Proposed Operating System for AeroNav Tech (ANT)

STUDENT ID: 5653630

INTRO

The autonomous drone market is forecast to grow exponentially, reaching $45.26 billion by 2028, at a compound annual growth rate (CAGR) of 21.5% (Research and Markets, 2024). This growing global demand highlights the need for AeroNav Tech's (ANT) proposed Operating System (OS) to integrate advanced features tailored to intelligent drones while prioritising safety, security, and reliability. Applications in modern drones range from filmmaking, offering cost-effective aerial perspectives (Vakhtin, 2023), to military operations, where they are transforming modern warfare through reconnaissance, intelligence gathering, and logistical tasks (Jacoby, 2024). Beyond these fields, their demand is increasing in other industries, such as delivery services, where companies are exploring new technologies to reduce delivery times (Amazon, 2022). Addressing these diverse needs, the proposed OS will be built as an embedded system using a real-time kernel, ensuring deterministic performance for safety-critical tasks such as navigation, obstacle avoidance, and communication. Furthermore, seamless integration with sensors, cameras, and AI-driven systems will enable efficient data processing and support advanced autonomous functionalities (ideaForge, 2023.).

PROCESS MANAGEMENT

The Operating System (OS) will manage lifecycle events, including creation, termination, synchronisation and incorporate APIs to provide functionality similar to those used in modern OS.(Silberschatz et al., 2012, pp. 107–109)

- Fork() – Spawns a new process based on the parent process.
- Wait() – Temporarily halts parent process until a child process completes.
- Exec() – Replaces current process with a specified one.
- Exit() – Ends a process and releases its associated resources.

The OS will then transition between the 5 process states as shown in Figure 1.The responsiveness will then be improved further by implementing an interrupt mechanism to allow the CPU to save its current state and handle critical tasks via an Interrupt Service Routine (ISR) (Labrosse, 2002, pp. 62, 64) ensuring efficient recovery during critical operations. Additionally, message passing will be used as the Interprocess Communication (IPC) method to ensure reliable communication with predictable execution times, isolating safety-critical tasks from non-critical ones to prevent interference and maintain system integrity(Labrosse, 2002, p. 66).

PROCESS MANAGEMENT SECURITY

To prevent malicious processes from being run (e.g., a process attempting to override navigation commands and causing the drone to deviate from its flight path),the OS will ensure that processes operate in user mode, restricting access to higher-privilege actions (Arpaci-Dusseau and Arpaci-Dusseau, 2014, pp. 46–47),since dynamic priority adjustment ensures critical tasks like obstacle detection are prioritised.

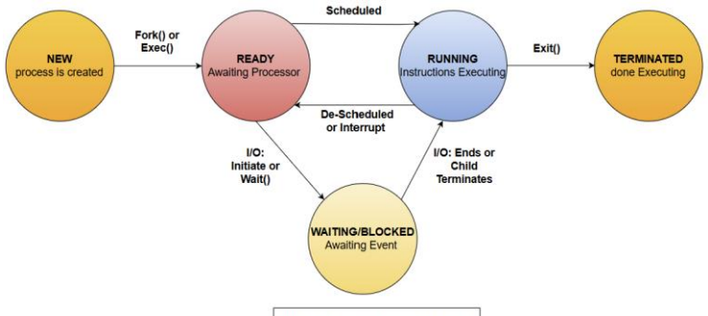
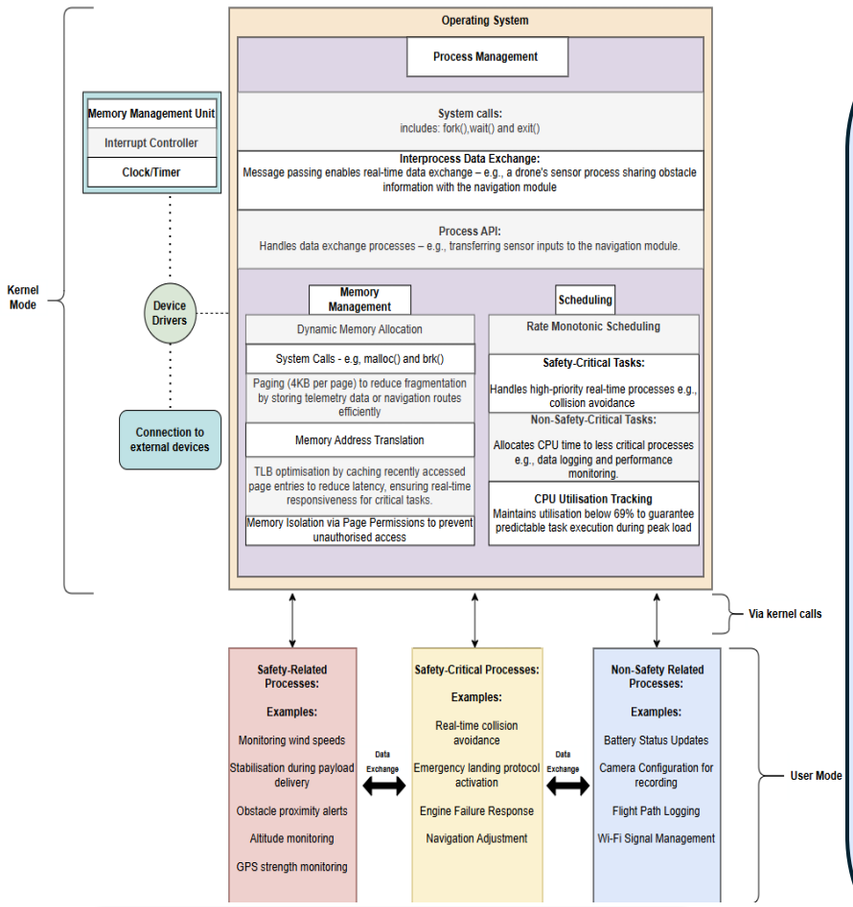


Figure 1: The Process States



SCHEDULING:

The OS will take inspiration from a Real-Time Operating System (RTOS) to ensure deterministic behavior and meet the real-time requirements of critical tasks. Within the RTOS, the Rate Monotonic Scheduling (RMS) algorithm will be implemented to prioritise periodic tasks based on their frequency.

RMS assigns higher priorities to tasks with shorter deadlines, ensuring timely execution of safety-critical tasks. The scheduler ensures that high-priority tasks, like collision avoidance and stabilisation, are executed first, while lower-priority tasks, such as data logging, are executed as resources allow.

To address potential drawbacks, such as the risk of starvation for lower-priority tasks, periodic priority boosts will be applied to ensure fairness. Moreover, by planning resource usage and limiting CPU utilisation below 69%, ensures that all critical tasks meet their deadlines by allocating sufficient CPU time to high-priority periodic tasks like collision avoidance and stabilisation.

MEMORY MANAGEMENT

The OS will utilise paging over segmentation to reduce both internal and external fragmentation. It will reduce external fragmentation through the use of page tables. To reduce internal fragmentation the OS will use small sized pages (about 4KB), since processes often require memory allocations that do not perfectly fit into a single page, smaller page sizes ensure that the unused space within each allocated page is reduced (Silberschatz et al., 2018, pp. 368).The OS will also keep a track of virtual pages by mapping them to its physical location with a unique page table for each process (Arpaci-Dusseau and Arpaci-Dusseau, 2014, p. 183). The OS will then make use of Translation-Lookaside Buffers (TLBs) to further optimise address translation, by caching recently accessed page table entries to reduce latency during virtual-to-physical address mapping, ensuring efficient memory management for real-time drone operations, such as stabilising flight during sudden wind changes or recalculating routes in response to no-fly zone restrictions (Arpaci-Dusseau and Arpaci-Dusseau, 2014, pp. 184, 187).

MEMORY MANAGEMENT SECURITY

The OS will implement memory isolation to ensure the security for critical processes, preventing unauthorised access or interference between processes e.g. isolating firmware updates from active flight operations. Translation-Lookaside Buffers (TLBs) will also be used to enforce memory access permissions during address translation, ensuring restricted memory regions remain secure e.g. blocking unauthorised processes from accessing encrypted communication data (Arpaci-Dusseau and Arpaci-Dusseau, 2014, pp. 183, 187). System calls such as `Fork()` to create new processes, `Exec()` to replace a process's memory space, and `Wait()` to synchronise operations will be validated by the OS to ensure they originate from authorised processes, preventing unauthorised control or disruption (Arpaci-Dusseau and Arpaci-Dusseau, 2014, p. 46).

CONCLUSION:

Overall, the proposed solution integrates advanced features such as RMS scheduling, paging, and memory isolation to ensure real-time responsiveness and enhanced security. Designed to address the unique challenges of drone operations, it prioritises safety-critical tasks like navigation and collision avoidance while maintaining system reliability. This design balances innovation and practicality, offering a scalable solution for the growing demand for autonomous systems across various industries

REFERENCES:

Research and Markets (2024). *Autonomous Drones Market Report 2024: Market Size and Trends*. Available at: https://www.researchandmarkets.com/report/autonomous-drone-market#:~:text=This%20Autonomous%20Drones%20market%20report%20covers%20market%20characteristics%2C,market%E2%80%99s%20historic%20and%20forecast%20market%20growth%20by%20geography. [Accessed: 8 January 2025].

Vakhtin, D. (2023). *Drone technology: Revolutionizing film budgeting*. Available at: https://filmustage.com/blog/drone-technology-revolutionizing-film-budgeting/ [Accessed 29 December 2024].

New York Post Article: Jacoby, T., 2024. AI is reshaping drone warfare in Russia and Ukraine.*New York Post*, 29 September. Available at: https://nypost.com/2024/09/29/world-news/ai-is-reshaping-drone-warfare-in-russian-and-ukraine/ [Accessed: 29 December 2024].

Amazon Article: Amazon Staff, 2022. How Amazon is building its drone delivery system. *About Amazon*. Available at: https://www.aboutamazon.com/news/transportation/how-amazon-is-building-its-drone-delivery-system [Accessed: 29 December 2024].

ideaForge. (2023). What's Inside Your Drone—The Power of Embedded Systems! Available at: https://ideaforgetech.com/blogs/what-is-inside-your-drone [Accessed: 6 January 2025]

Silberschatz, A., Galvin, P.B., and Gagne, G. (2012). Operating System Concepts. 9th edn. Hoboken, NJ: John Wiley & Sons.

Labrosse, J.J. (2002). MicroC/OS-II: The Real-Time Kernel. 2nd edn. Burlington, MA: CMP Books.

Arpaci-Dusseau, R.H. and Arpaci-Dusseau, A.C. (2014). Operating Systems: Three Easy Pieces. 1st edn. Arpaci-Dusseau Books, LLC.