|  |  |  |
| --- | --- | --- |
| Critère | PlatformIO Core | STM32Cube |
| Flexibilité Matérielle | Compatible avec plusieurs MCU (STM32, ESP32) | Spécifique aux microcontrôleurs STM32 |
| IDE/Interface | Fonctionne avec VS Code (ergonomique) | STM32CubeIDE (moins modulable) |
| Support FreeRTOS | Très bien intégré | Intégration prête à l'emploi |
| Bibliothèques | Gestionnaire riche et facile d'accès | Axé sur les outils STM32 spécifiques |
| Facilité d'utilisation | Approche intuitive et flexible | Besoin d'apprendre l'écosystème STM32 |
| Communauté | Large et active (ESP32 et STM32 inclus) | Spécifique aux utilisateurs STM32 |
| Adaptabilité | Transition facile entre différents MCU | Adapté uniquement aux projets STM32 |

Plateforme de Développement : PlatformIO vs STM32Cube

Matériel : STM32 vs ESP32

|  |  |  |
| --- | --- | --- |
| Critère | PlatformIO Core | STM32Cube |
| Performances | Large gamme (du bas au très haut de gamme) | Performances correctes, mais limitées |
| Connectivité | Nécessite des modules additionnels | Intégré (Wi-Fi, Bluetooth) |
| Robustesse | Fiable, idéal pour des applications critiques | Moins robuste, idéal pour des prototypes |
| Documentation | Documentation riche et officielle | Documentation riche avec une grande communauté |
| Coût | Plus cher | Abordable |
| Facilité d'utilisation | Nécessite des configurations plus complexes | Plus facile à prendre en main |
| Applications | Industrielles, exigeantes | Prototypes, solutions connectées |

A close-up of a computer chip

Description automatically generated

Comparatif des RTOS

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RTOS | Léger | Facilité d'Utilisation | Multithreading | Support STM32 | Support ESP32 | Écosystème Avancé | Licence |
| FreeRTOS | Oui | Facile | Oui | Oui | Oui | Basique | Libre (MIT) |
| Zephyr | Non | Moyenne | Oui | Oui | Oui | Avancé | Libre (Apache 2.0) |
| RT-Thread | Oui | Moyenne | Oui | Oui | Oui | Moyen | Libre (Apache 2.0) |
| ChibiOS | Oui | Facile | Oui | Oui | Oui | Basique | Libre (GPL3) |
| Micrium OS | Non | Moyenne | Oui | Oui | Oui | Avancé (industriel) | Commercial |
| RIOT OS | Oui | Moyenne | Oui | Oui | Oui | Moyen | Libre (LGPL) |

Système d’Exploitation Temps Réel : FreeRTOS

|  |  |
| --- | --- |
| Critère | FreeRTOS |
| Poids | Très léger, optimisé pour MCU |
| Compatibilité | Support natif sur STM32 et ESP32 |
| Multitâche | Gestion des tâches, priorités, mutex, etc. |
| Documentation | Riche et bien structurée |
| Communauté | Large avec des exemples nombreux |
| Complexité | Facile à intégrer avec les bonnes ressources |

|  |  |  |
| --- | --- | --- |
| Critères | C | C++ |
| Modularité | Basique : nécessite des structures et des fonctions manuelles. | Avancée : classes, objets, interfaces facilitent l’organisation du code. |
| Complexité du Code | Simple, mais devient difficile à gérer pour des systèmes complexes. | Gestion facile des systèmes complexes grâce à l’orienté objet. |
| Gestion des Capteurs/Moteurs | Doit être gérée manuellement avec des structures. | Encapsulation avec des classes pour les capteurs et moteurs. |
| Support de Simulation (Unity/Gazebo) | Possible, mais nécessite des wrappers complexes. | Meilleur support grâce aux bibliothèques et aux objets. |
| Efficacité | Très rapide et léger, parfait pour des systèmes simples. | Légèrement moins rapide à cause des abstractions, mais toujours performant. |
| Évolutivité | Limitée : ajouter de nouvelles fonctionnalités peut être compliqué. | Haute : ajout facile de fonctionnalités avec polymorphisme et héritage. |
| Temps Réel (RTOS) | Bien supporté (FreeRTOS, Zephyr). | Très bien supporté (FreeRTOS, Zephyr) avec des abstractions supplémentaires. |
| Facilité d’Apprentissage | Plus simple pour débuter. | Plus complexe, mais offre plus d’outils et de flexibilité. |
| Compatibilité avec Unity | Doit être intégré via des bibliothèques C# ou wrappers simples. | Meilleure intégration grâce à des bibliothèques et API avancées. |
| Bibliothèques Standard | Limité : pas de bibliothèque standard pour des fonctionnalités avancées. | Très riche (STL : vecteurs, files d’attente, etc.). |
| Licence Logicielle | Libre, souvent utilisé dans des environnements embarqués. | Libre, avec des extensions avancées possibles pour les systèmes modernes. |

|  |  |
| --- | --- |
| Utilisez C++ si : | Utilisez C si : |
| - Vous avez besoin de modularité et de scalabilité. | - Le système est très contraint en mémoire et CPU. |
| - Vous souhaitez une intégration facile avec Unity ou Gazebo. (simulateur) | - Le projet reste très simple (pas de simulation ou peu d’évolution). |
| - Vous prévoyez d’ajouter des fonctionnalités avancées ou de tester plusieurs algorithmes. | - Vous privilégiez la légèreté et la simplicité. |

Résumé des Recommandations

|  |  |
| --- | --- |
| Catégorie | Recommandation |
| Plateforme | PlatformIO pour flexibilité et modularité  Si on veut changer de microcontrôleurs est compatible avec plusieurs + transition facile |
| Matériel | STM32 pour robustesse ou ESP32 pour prototypes |
| RTOS | FreeRTOS pour multitâche et gestion en temps réel |
| Langage | C++ |

Simulation

<https://unity.com/fr/blog/engine-platform/robotics-simulation-is-easy-as-1-2-3>