# INTEGRAÇÃO DO MICRO-ROS COM ESP32-S3 E APLICAÇÕES NA ROBÓTICA MÓVEL

MADANA N'BANA[1]; ISAAC VICTOR CAMPELO DE SOUZA[2]; LORENZZO VILELA DA SILVA[3]; DANIELLE MARTINS DECKER[4]

MARCELO LEMOS ROSSI[5]

*[1]Universidade Federal de Pelotas – madananbana5@gmai.com*
*[2] Universidade Federal de Pelotas – isaacvcampelo@gmail.com*
*[3] Universidade Federal de Pelotas – lezandrosw@gmail.com*
*[4] Universidade Federal de Pelotas – daniellemrtnsdecker@gmail.com*
*[5] Universidade Federal de Pelotas – marcelo.rossi@ufpel.edu.br*

## 1. INTRODUÇÃO

A robótica móvel vem se expandindo com a crescente demanda por veículos autônomos, drones, robôs de entrega e plataformas de vigilância capazes de operar em ambientes complexos. Para aplicações embarcadas de baixo custo e alta eficiência, a integração de sistemas operacionais leves com middlewares robustos de comunicação tem se tornado uma necessidade. O Micro-ROS, uma adaptação do ROS 2 (Robot Operating System), surge como uma solução ideal para microcontroladores como o ESP32-S3, que possui capacidade computacional e conectividade Wi-Fi e Bluetooth integradas.

## 2. METODOLOGIA

Neste trabalho, utilizamos ROS 2 e Micro-ROS para integração e controle de sistemas embarcados.

O Micro-ROS é uma implementação do ROS 2 para ser executada por sistemas embarcados. Ela é baseada no protocolo XRCE-DDS (*eXtremely Resource Constrained Environments* DDS), uma versão leve do padrão DDS (*Data Distribution Service*) voltada para dispositivos com recursos computacionais limitados (RAM, processamento e armazenamento).

A comunicação entre o microcontrolador e o ROS 2 se dá por meio de um agente, o Micro-ROS Agent, que atua como ponte entre os nós embarcados e os demais nós ROS 2 do sistema (MICRO-ROS, 2025). *A arquitetura do Micro-ROS divide-se entre o nó Agente e o nó cliente*.

O nó agente (Micro-ROS Agent) roda em um computador tradicional ou computador de placa única (SBC, do inglês *Single Board Computer*) como, por exemplo, uma Raspberry Pi. Atualmente, nós estamos rodando em um computador para o desenvolvimento e testes. Já o nó cliente (Micro-ROS Client) roda em um microcontrolador, enviando dados e recebendo dados do agente via serial, Wi-Fi, CAN ou outras formas de comunicação. No nosso caso estamos utilizando a ESP32-S3 como microcontrolador e enviando os dados via Wi-Fi utilizando UDP.

O ESP32-S3 é um microcontrolador de 32 bits desenvolvido pela Espressif, baseado no processador Xtensa LX7 de dois núcleos de processamentos e possui conectividade Wi-Fi, sendo adequado para aplicações que requerem processamento paralelo e conectividade sem fio (ESPRESSIF, 2024). Seu uso com

a ESP-IDF (*Espressif IoT Development Framework*) o torna compatível com a maioria dos componentes ROS via Micro-ROS.

## 3. RESULTADOS E DISCUSSÃO

Podemos separar o processo de integração do Micro-ROS com ESP32-S3 em requisitos do ambiente de desenvolvimento e em configuração do projeto.

Como requisitos do ambiente de desenvolvimento temos que a integração do Micro-ROS com o ESP32-S3 exige:

- ESP-IDF versão >= 5.0 e < 5.4;
- Ferramentas de build (CMake, Ninja e Python3);
- Extensão (pacotes do Micro-ROS como componente;
- Micro-ROS Agent instalado no host (Linux ou Windows via WSL);
- Interface de comunicação (UART/Wi-Fi ou utras).

Para a configuração do projeto deve-se seguir a estrutura padrão da ESP-IDF, com a criação de um diretório de aplicação e adição do componente Micro-ROS. A comunicação com o agente pode ocorrer via UART, ou Wi-Fi, ou outra interface de comunicação, porém a comunicação por outras interfaces exige a implementação da interfase no código que via embarcado na ESP32-S3. O código principal realiza a inicialização do sistema com chamadas como rmw_uros_set_custom_transport() e rclc_executor_spin() (MICRO-ROS, 2023).

Para a configuração do ambiente de desenvolvimento deve ser feita seguindo os passos:

1. Instalação da ESP-IDF: Utiliza-se o instalador ou terminal. Após a configuração, o comando idf.py create-project inicia um novo projeto.
2. Adição do Micro-ROS como submódulo Git (dependêmicas):
   - cd componentes
   - git clone -b foxy https://github.com/micro-ROS/micro_ros_espidf_component.git
   - micro_ros_espidf_component
3. Configuração com menuconfig: idf.py menuconfig
   - Nessa etapa, configura-se a porta serial, o transport layer (ex: Wi-Fi), e parâmetros do agente.
4. Código de inicialização:
   - No app_main() é necessário iniciar o rmw_uros_set_custom_transport() com o transporte UART e inicializar o nó com rcl_node_init.
5. Compilação e Flash:
   - idf.py build
   - idf.py flash monitor
6. Execução do Agente Micro-ROS no computador host: No computador hospedeiro (host) ou Raspberry Pi, executa-se o agente com o seguinte comando:
   - ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/ttyUSB0

Esse agente conecta-se ao microcontrolador e o integra à rede ROS 2, permitindo a troca de mensagens padronizadas, sendo essencial garantir a compatibilidade de bibliotecas, versões e memória disponível. Além disso,

recomenda-se a utilização de FreeRTOS, que já vem integrado à ESP-IDF, para garantir multitarefa e tempo real.

O procedimento apresentado acima tem várias aplicações na robótica móvel, possibilitando o desenvolvimento de robôs mais flexíveis. Por exemplo, na navegação autônoma: Com o Micro-ROS, o ESP32-S3 pode atuar como nó sensor de um robô móvel, publicando dados de sensores como IMU, ultrassônicos e odometria, e recebendo comandos de navegação de um nó mestre no ROS 2. Essa abordagem reduz o processamento a bordo e distribui as tarefas entre nós inteligentes.

Drones podem utilizar o ESP32-S3 com Micro-ROS para monitoramento de sensores e envio de dados de telemetria ao ROS 2 via Wi-Fi. Isso é útil para mapeamento, patrulhamento ou inspeção aérea, especialmente em ambientes com limitação de peso e energia.

Robôs Swarm (Enxame) é outra aplicação, neste caso voltado à robótica em enxame, onde múltiplos ESP32-S3 comunicam-se via Micro-ROS para coordenar ações em grupo, como varredura de áreas, seguindo topologias como a malha (mesh) ou em estrela, com um agente central.

Conforme apresentado a integração Micro-ROS permite que a ESP32-S3 atue como um nó ROS 2 remoto, publicando dados de sensores e subscrevendo comandos de controle em tópicos como /imu/data e /odom, e recebe comandos de controle como velocidade linear e angular pelo tópico /cmd_vel permitindo controle baseado em SLAM ou navegação autônoma. Isso possibilita o uso de ferramentas ROS 2 como RViz, SLAM Toolbox e navegação autônoma (GUTIERREZ et al., 2020).

No caso deste projeto estamos utilizando o Micro-ROS para controlar o carro autônomo utilizando sensores (IMU, encoder, ultrassônicos, etc.) para controlar velocidade e direção, comunicação com estação base ROS 2 (como navegação em linha, desvio de obstáculos).

A aplicação desta metodologia tem permitido Redução de custo e consumo energético; Maior modularidade e escalabilidade para sistemas multi-robôs; Reuso de ferramentas ROS 2, como RViz, rqt e rosbag; Padronização da comunicação e interoperabilidade com bibliotecas ROS 2

## 4. CONCLUSÕES

A integração do Micro-ROS com o ESP32-S3 representa um avanço significativo na robótica móvel de baixo custo, permitindo que microcontroladores participem ativamente de ecossistemas ROS 2.

Futuramente, espera-se maior integração com sensores embarcados e arquiteturas de inteligência artificial.

## 5. REFERÊNCIAS BIBLIOGRÁFICAS

ESPRESSIF. **ESP32-S3 Technical Reference Manual**. 2024. Disponível em: https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3. Acesso em: 21 jul. 2025.

ESPRESSIF DOCS. G**etting Started with ESP-IDF**. 2024. Disponível em: https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/get-started/. Acesso em: 21 jul. 2025.

MARUYAMA, Y. et al. **Exploring the performance of ROS2**. In: Proceedings of the 13th International Conference on Embedded Software. EMSOFT '16. New York: ACM, 2016. Disponível em: https://dl.acm.org/doi/10.1145/2968478.2968502. Acesso em: 21 jul. 2025.

MICRO-ROS. **Micro-ROS official documentation**. 2023. Disponível em: https://micro.ros.org/. Acesso em: 25 jul. 2025.

GUTIERREZ, R. A. et al. **Micro-ROS:** bridging the gap between embedded and general-purpose robotic systems. In: ROSCon 2020. Virtual: Open Robotics, 2020. Disponível em: https://micro.ros.org/docs/papers/roscon2020/. Acesso em: 25 jul. 2025.