



---

# Exploring Real-Time WebRTC Interface on MCUs & MPUs for Unitree GO2 Robots

---

Akshara Kuduvalli

# Motivation and Objectives

---

## Motivation:

- There is one main Unitree GO2 Web-based Communication Interface using WebRTC over ROS2, but is designed to run on a full computer
  - No microcontroller-based solution for Unitree Go2 Robot control

## Objective:

- Explore building a lightweight WebRTC client (only data channel, not audio and video) on an MCU to communicate in real-time with Unitree Go2 Robots
- Opens up the possibility to communicate (sensor acquisition, actuation) with the GO2 robot across many different edge devices.



# Technical Approach and Novelty

---

Unlike past web-based communication implementations to GO2 robots, there is little information online about establishing a WebRTC client on a MCU rather than a full microprocessor

- Robot controlled through full WebRTC stacks (Python/ROS2, Unitree desktop software).
- Not suitable for microcontrollers due to resource limits & general lack of existing WebRTC stack implementations.

## Novel Approach:

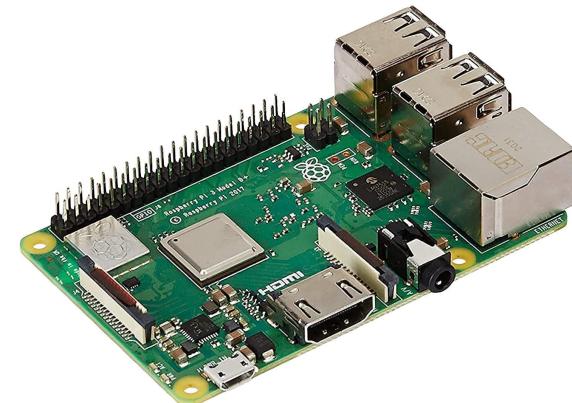
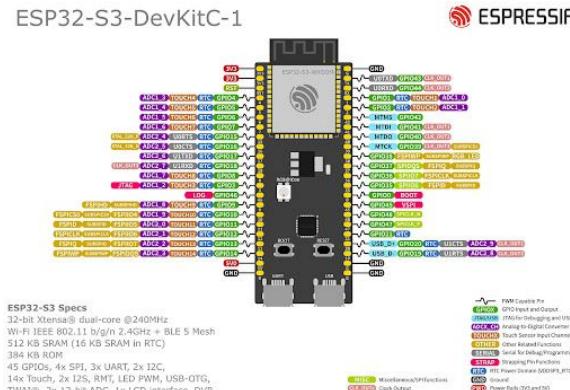
- Implement a light WebRTC transport layer on an ESP32-S3 running Zephyr, designed specifically for data transfer with GO2 Robot
- Handle Wi-Fi, event callbacks, and real-time packet generation using Zephyr threads.

### go2-webrtc - WebRTC API or Unitree GO2 Robots



# Methods

- **ESP32-S3-N16R8 board**
  - built-in Wi-Fi with additional Antenna, 8MB PSRAM, 16MB Flash
- **Built on the Zephyr RTOS platform**
  - Robust scheduling, existing ESP32s3 driver support, WiFi mgmt, DTLS, WebSockets
- **Exploring building the RTCDataChannel as part of the WebRTC interface** (for data transfer)
- Also exploring implementing existing full WebRTC stack on Raspberry Pi as backup platform



# Evaluation and Metrics

---

## Functional Metrics

- Successful Wi-Fi connection & IP acquisition
- Ability to open a data channel to robot (whether on MCU/raspberry pi)
- Successful bidirectional message exchange

## Performance Metrics

- Measure end-to-end command latency (in milliseconds)
- Packet loss rate on WiFi
- Stability over time

## Resource Metrics

- CPU load on ESP32-S3 under an RTOS
- Memory footprint from network stacks & buffers

## Success Criteria

- Robot responds correctly to embedded-generated commands
- Telemetry received continuously

# Current Status and Next Steps

---

Zephyr toolchain and WiFi connectivity verified on ESP32-S3.

Successful event-based WiFi connection and IP acquisition using Zephyr net\_event and wifi\_mgmt APIs

Established WebSocket data transfer as client

Established WebSocket and WiFi Connectivity on ESP32-s3 (Done)

UDP Socket on Zephyr

Simplified SCTP-like framing for control packets

Control Packet Encoding

RTCDATAChannel over DTLS for data transfer

Test with real Unitree GO2 Robot to establish sending and receiving data commands

(thanks to help from Julian de Gortari Briseno from NESL)

TODO: live testing

Exploring WebRTC layer in Zephyr on ESP32-s3 , also on raspberry pi in python (in progress)