MMI Project: Controlling a LED over ROS2 with Keyboard inputs on Windows 11 through WSL2

Michael Smirnov - TU Wien

December 14, 2024

The current project deviated significantly from the initial proposal, which was more ambitious in its aim to establish a connection between an AR headset and the hardware. However, due to the lack of Wi-Fi support and time constraints, this initial objective could not be achieved. A feasible approach would involve hosting a simple and lightweight web server to receive TCP packets sent by a Unity application using the ros_unity_integration library [Unity(2022)] and forwarding them into the ROS2 system for processing by individual nodes. Alternatively, routing TCP packets from a Unity application running on Windows 11 to WSL2 proved to be configuration-intensive and not particularly straightforward. For future projects, adopting a native Unix development environment is recommended to simplify the process. Additionally, using a WiFi-enabled microcontroller, such as an ESP32 or a Raspberry Pi connected to an Arduino, could provide a more efficient and practical solution.

Overview

This project demonstrates a ROS2-based system for controlling RGB LEDs on an Arduino via keyboard input. The system architecture comprises three primary ROS2 nodes, a custom message type, and an Arduino device connected through serial communication. Nodes exchange data over two dedicated topics, enabling a closed control loop where user input leads to LED color changes that are subsequently reported back from the Arduino to the ROS2 environment.

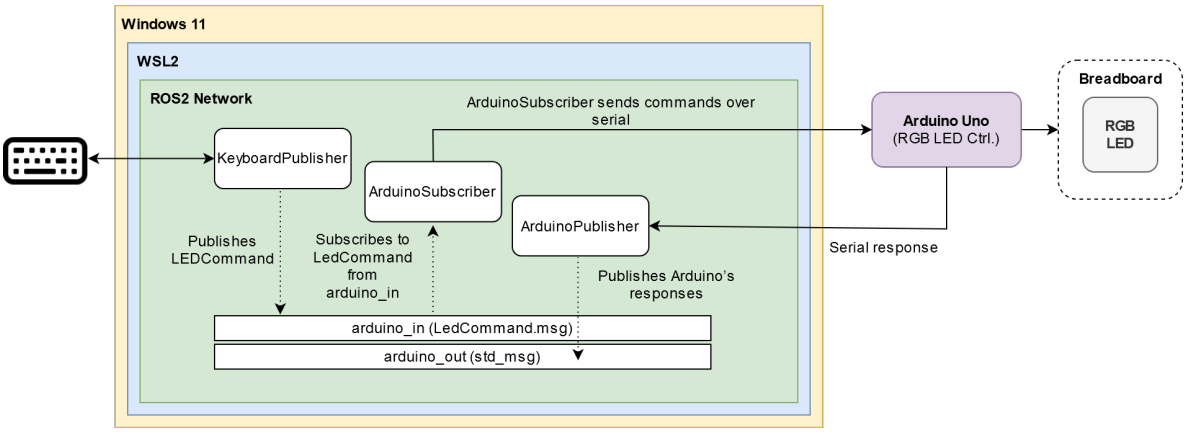

Figure 1: System Diagram

System Components and Data Flow

ROS2 Topics

- **arduino_in**: Receives `LedCommand` messages specifying an LED ID and corresponding RGB values. The `KeyboardPublisher` node posts these commands, and the `ArduinoSubscriber` node consumes them.

- **arduino_out**: Receives status updates from the Arduino via the `ArduinoPublisher` node. Messages published here confirm which commands the Arduino has executed.

ROS2 Nodes

- **KeyboardPublisher**: Monitors keyboard input from the user's console. For each valid input, it constructs and publishes a `LedCommand` message on `arduino_in`.
- **ArduinoSubscriber**: Subscribes to `arduino_in`, and on receiving a `LedCommand`, sends the corresponding instructions to the Arduino over a serial connection (e.g., `/dev/ttyACM0`). This node translates ROS2 commands into C instructions for the Arduino.
- **ArduinoPublisher**: Reads responses from the Arduino's serial output and publishes them to `arduino_out`. These responses confirm processed commands or report the currently set LED states.

Custom Message Type

- **LedCommand.msg**: Defines:

  led_id Identifies which LED to control.

  r_value, g_value, b_value Specify the desired red, green, and blue intensity values.

Arduino Setup and Code

The Arduino runs a custom sketch that:

- Receives incoming serial commands specifying LED IDs and RGB values.
- Updates the LED's color output accordingly.
- Sends a confirmation or status message back over serial, which the `ArduinoPublisher` node then publishes to `arduino_out`.

WSL2 and USB Integration

For environments running under WSL2, USB devices are passed through using `usbipd-win`. This allows the Arduino to appear as a serial device inside WSL2. Developers can:

1. Bind and attach the Arduino's USB connection to WSL2.
2. Flash the Arduino from Windows, then reattach it to WSL2 for runtime use.

Resulting Behavior

When a user presses a key, `KeyboardPublisher` transforms that input into a color command (e.g., changing an LED from red to green). The `ArduinoSubscriber` node forwards this command to the Arduino, which updates the LED and returns a confirmation message. The `ArduinoPublisher` node then publishes this confirmation into the ROS2 environment, completing the feedback loop.

This design separates user input handling, device communication, and feedback reporting, making the system both modular and transparent.

References

[Unity(2022)] Unity. ros_unity_integration, 2022. URL https://github.com/Unity-Technologies/Unity-Robotics-Hub/tree/main/tutorials/ros_unity_integration. Accessed: 2024-12-14.