# Complete ESP32 WROOM + ROS2 Humble Setup Documentation

## Ubuntu 24.04 - Step by Step with All Errors and Solutions

**Hardware:** ESP32 WROOM DevKit
**Port Detected:** /dev/ttyACM0
**OS:** Ubuntu 24.04 LTS
**ROS2:** Humble Hawksbill
**Date:** October 2025

---

## Table of Contents



---

## PART 1: Fix Port Permissions

### Problem

Ubuntu 24.04 does not detect ESP32 port in Arduino IDE by default due to permission issues.

### Solution

**Step 1:** Add user to dialout group

bash

```bash
sudo usermod -a -G dialout $USER
```

**Step 2:** Add user to tty group

bash

```
sudo usermod -a -G tty $USER
```

**Step 3:** Remove conflicting brltty package

bash

```bash
sudo apt remove brltty -y
```

**Step 4:** Reload udev rules

bash

```bash
sudo udevadm control --reload-rules
sudo udevadm trigger
```

**Step 5:** ⚠️ CRITICAL - Reboot system

bash

```bash
sudo reboot
```

## Verification After Reboot

**Check port detection:**

bash

```bash
ls /dev/ttyUSB* /dev/ttyACM*
```

**Output:** `/dev/ttyACM0` ✅

**Verify group membership:**

bash

```
groups $USER
```

**Output:** `jittu : jittu adm tty dialout cdrom sudo dip plugdev lpadmin lxd sambashare` ✅

**Check system logs (optional - gave permission error):**

bash

```
dmesg | tail -20
```

**Error:** `operation not permitted`
**Solution:** Not critical, skipped this step

---

# PART 2: Install ROS2 Dependencies

## Step 1: Update system

bash

```
sudo apt update
```

## Step 2: Source ROS2 Humble

bash

```
source /opt/ros/humble/setup.bash
```

## Step 3: Install RMW CycloneDDS

bash

```
sudo apt install -y ros-humble-rmw-cyclonedds-cpp
```

**Status:** ✅ Installed successfully

### Step 4: Install micro-ROS setup

bash

```bash
sudo apt install -y ros-humble-micro-ros-setup
```

**Error:**

```
E: Unable to locate package ros-humble-micro-ros-setup
```

**Solution:** Package not available in apt, will build from source later in Part 4

### Step 5: Install colcon tools

bash

```bash
sudo apt install -y python3-colcon-common-extensions
```

**Status:** ✅ Installed successfully

### Step 6: Install Python serial

bash

```bash
sudo apt install -y python3-serial
```

**Status:** ✅ Installed successfully

### Step 7: Set RMW implementation

bash

```
export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp
```

### Step 8: Add RMW to bashrc

bash

```bash
echo "export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp" >> ~/.bashrc
```

---

## PART 3: Install Arduino IDE

### Step 1: Go to Downloads folder

bash

```bash
cd ~/Downloads
```

### Step 2: Download Arduino IDE 2.x

bash

```bash
wget https://downloads.arduino.cc/arduino-ide/arduino-ide_2.3.2_Linux_64bit.AppImage
```

### Step 3: Make executable

bash

```bash
chmod +x arduino-ide_2.3.2_Linux_64bit.AppImage
```

### Step 4: Install permanently (optional)

bash

```
./arduino-ide_2.3.2_Linux_64bit.AppImage --install
```

**Status:** ✅ Arduino IDE now available in applications menu

### Step 5: Configure ESP32 Board Support (IN ARDUINO IDE GUI)

1. Open Arduino IDE
2. **File → Preferences**
3. In "Additional Board Manager URLs" add:

```
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
```

4. Click **OK**
5. **Tools → Board → Boards Manager**
6. Search: **ESP32**
7. Install: **"esp32 by Espressif Systems"**
8. Close Arduino IDE

**Status:** ✅ ESP32 board support installed

---

## PART 4: Create ROS2 Workspace

### Step 1: Create workspace directory

bash

```bash
mkdir -p ~/ros2_ws/src
```

### Step 2: Navigate to src

bash

```bash
cd ~/ros2_ws/src
```

### Step 3: Clone micro-ROS setup (building from source due to apt error)

bash

```bash
git clone -b humble https://github.com/micro-ROS/micro_ros_setup.git
```

### Step 4: Navigate to workspace root

bash

```bash
cd ~/ros2_ws
```

### Step 5: Build workspace

bash

```bash
colcon build
```

**Output:**

```
Starting >>> micro_ros_setup
Finished <<< micro_ros_setup [0.09s]
Summary: 1 package finished
```

**Status:** ✅ Built successfully

### Step 6: Source workspace

bash

```bash
source install/setup.bash
```

### Step 7: Add to bashrc

bash

```bash
echo "source ~/ros2_ws/install/setup.bash" >> ~/.bashrc
```

---

# PART 5: Install micro-ROS Agent

### First Attempt (Failed)

bash

```bash
sudo apt install ros-humble-micro-ros-agent
```

**Error:**

```
E: Unable to locate package ros-humble-micro-ros-agent
```

### Solution: Build from Source

**Step 1:** Navigate to workspace src

bash

```bash
cd ~/ros2_ws/src
```

**Step 2:** Clone micro-ROS agent

bash

```bash
git clone -b humble https://github.com/micro-ROS/micro-ros-agent.git
```

**Step 3:** Navigate to workspace root

bash

```bash
cd ~/ros2_ws
```

**Step 4:** Build workspace

bash

```bash
colcon build
```

**Output:**

```
Starting >>> micro_ros_setup
Finished <<< micro_ros_setup [0.09s]
[Processing: micro_ros_agent]
--- stderr: micro_ros_agent
Cloning into 'xrceagent'...
HEAD is now at 57d0862 Release v2.4.2
CMake Warning (dev) at /usr/share/cmake-3.22/Modules/FindPackageHandleStandardArgs.cmake:438 (message):
  The package name passed to `find_package_handle_standard_args` (tinyxml2)
  does not match the name of the calling package (TinyXML2).
---
Finished <<< micro_ros_agent [32.3s]
Summary: 2 packages finished [32.4s]
  1 package had stderr output: micro_ros_agent
```

**Warning Analysis:** The CMake warning about tinyxml2 is harmless and can be ignored.

**Step 5:** Source workspace

bash

```
source install/setup.bash
```

**Status:** ✅ micro-ROS agent built successfully

---

# PART 6: Setup Arduino IDE for ESP32

## Step 1: Open Arduino IDE

bash

```
cd ~/Downloads
./arduino-ide_2.3.2_Linux_64bit.AppImage
```

**OR** Open from applications menu (if installed permanently)

## Step 2: Install micro-ROS library (IN ARDUINO IDE)

1. **Sketch → Include Library → Manage Libraries**
2. Search: **micro_ros_arduino**
3. Find: **"micro_ros_arduino by micro-ROS"**
4. Click **Install**
5. Wait for installation to complete

**Status:** ✅ Library installed

## Step 3: Select Board

- **Tools → Board → ESP32 Arduino → ESP32 Dev Module**

## Step 4: Select Port

- **Tools → Port → /dev/ttyACM0**

## Step 5: Configure Upload Speed

- **Tools → Upload Speed → 115200**

**Configuration Complete:** ✅

---

# PART 7: Upload Code to ESP32

## ESP32 Test Code

**File → New Sketch**, then paste this code:

cpp

```
#include <micro_ros_arduino.h>
#include <stdio.h>
#include <rcl/rcl.h>
#include <rcl/error_handling.h>
#include <rclc/rclc.h>
#include <rclc/executor.h>
#include <std_msgs/msg/string.h>

rcl_publisher_t publisher;
std_msgs__msg__String msg;
rclc_executor_t executor;
rclc_support_t support;
rcl_allocator_t allocator;
rcl_node_t node;
rcl_timer_t timer;

#define LED_PIN 2

void error_loop() {
  while(1) {
    digitalWrite(LED_PIN, !digitalRead(LED_PIN));
    delay(100);
  }
}

void timer_callback(rcl_timer_t * timer, int64_t last_call_time) {
  if (timer != NULL) {
    sprintf(msg.data.data, "Hello from ESP32");
    msg.data.size = strlen(msg.data.data);
    rcl_publish(&publisher, &msg, NULL);
    digitalWrite(LED_PIN, !digitalRead(LED_PIN));
  }
}

void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(115200);
  delay(2000);

  set_microros_transports();
  allocator = rcl_get_default_allocator();
```

```
  rclc_support_init(&support, 0, NULL, &allocator);
  rclc_node_init_default(&node, "esp32_node", "", &support);

  rclc_publisher_init_default(
    &publisher, &node,
    ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, String),
    "esp32_chatter");

  rclc_timer_init_default(&timer, &support, RCL_MS_TO_NS(1000), timer_callback);
  rclc_executor_init(&executor, &support.context, 1, &allocator);
  rclc_executor_add_timer(&executor, &timer);

  msg.data.data = (char *)malloc(100 * sizeof(char));
  msg.data.size = 0;
  msg.data.capacity = 100;
}

void loop() {
  rclc_executor_spin_some(&executor, RCL_MS_TO_NS(100));
  delay(100);
}
```

### Upload Process

1. Ensure ESP32 is connected via USB to `/dev/ttyACM0`
2. Click **Upload (→)** button
3. Wait for compilation and upload

**Status:** ✅ Done uploading

---

# PART 8: Test Communication

## Terminal 1: Start micro-ROS Agent

### First Attempt (with RMW_IMPLEMENTATION set):

bash

```
source /opt/ros/humble/setup.bash
source ~/ros2_ws/install/setup.bash
ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/ttyACM0 -b 115200
```

**Output:**

```
[1759388275.916129] info | TermiosAgentLinux.cpp | init | running... | fd: 3
[1759388275.916346] info | Root.cpp | set_verbose_level | logger setup | verbose_level: 4
[1759388275.916681] info | Root.cpp | create_client | create | client_key: 0x1DA0A926, session_id: 0x81
[1759388275.916712] info | SessionManager.hpp | establish_session | session established | client_key: 0x1DA0A926, address: 0
[1759388275.941180] info | ProxyClient.cpp | create_participant | participant created | client_key: 0x1DA0A926, participant_id: 0x000(1)
[1759388275.955473] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x1DA0A926, topic_id: 0x000(2), participant_id: 0x000(1)
[1759388275.965099] info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x1DA0A926, publisher_id: 0x000(3), participant_id: 0x000(1)
[1759388275.976185] info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x1DA0A926, datawriter_id: 0x000(5), publisher_id: 0x000(3)
```

**Agent Status:** ✅ Running and connected to ESP32

## Terminal 2: Check Topics (First Attempt - FAILED)

bash

```
source /opt/ros/humble/setup.bash
source ~/ros2_ws/install/setup.bash
ros2 topic list
```

**Problem:** Topics not showing up
**Issue:** RMW implementation mismatch between agent and ROS2 commands

## Solution: Unset RMW Implementation

**Terminal 1:** Stop agent (Ctrl+C), then:

bash

```bash
unset RMW_IMPLEMENTATION
ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/ttyACM0 -b 115200
```

**Terminal 2:**

bash

```bash
unset RMW_IMPLEMENTATION
unset ROS_DOMAIN_ID
source /opt/ros/humble/setup.bash
ros2 topic list
```

**After ESP32 Reset:** Topics still not visible

## Final Solution: Reset ESP32 After Agent Start

1. Start agent in Terminal 1 (without RMW_IMPLEMENTATION)
2. **Press RESET button on ESP32** (or unplug/replug USB)
3. Agent shows new connection messages
4. In Terminal 2, run:

bash

```bash
ros2 topic list
```

**Output:** `/esp32_chatter` ✅ **TOPIC VISIBLE!**

## Test Message Reception

bash

```bash
ros2 topic echo /esp32_chatter
```

**Output:**

```
data: Hello from ESP32
---
data: Hello from ESP32
---
```

**Status:** ✅✅✅ **WORKING!**

### Verify Node

bash

```bash
ros2 node list
```

**Output:** `/esp32_node` ✅

### Check Topic Info

bash

```bash
ros2 topic info /esp32_chatter
```

**Output:**

```
Type: std_msgs/msg/String
Publisher count: 1
Subscription count: 1
```

---

## PART 9: Troubleshooting Summary

### Issue 1: Port Not Detected

**Symptoms:** Arduino IDE doesn't show ESP32 port
**Cause:** User not in dialout/tty groups

**Solution:**

bash

```bash
sudo usermod -a -G dialout $USER
sudo usermod -a -G tty $USER
sudo reboot
```

## Issue 2: Package Not Found (micro-ros-setup)

**Error:** `E: Unable to locate package ros-humble-micro-ros-setup`
**Cause:** Package not available in Ubuntu 24.04 apt repositories
**Solution:** Build from source

bash

```bash
cd ~/ros2_ws/src
git clone -b humble https://github.com/micro-ROS/micro_ros_setup.git
cd ~/ros2_ws
colcon build
```

## Issue 3: Package Not Found (micro-ros-agent)

**Error:** `E: Unable to locate package ros-humble-micro-ros-agent`
**Cause:** Package not available in Ubuntu 24.04 apt repositories
**Solution:** Build from source

bash

```bash
cd ~/ros2_ws/src
git clone -b humble https://github.com/micro-ROS/micro-ros-agent.git
cd ~/ros2_ws
colcon build
```

## Issue 4: Topics Not Visible in ROS2

**Symptoms:**

- Agent shows ESP32 connected
- `ros2 topic list` shows no topics

- `ros2 node list` shows no nodes

**Cause:** RMW implementation mismatch between micro-ROS agent and ROS2
**Solution:**

bash

```bash
# In both terminals
unset RMW_IMPLEMENTATION
unset ROS_DOMAIN_ID

# Terminal 1: Start agent
ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/ttyACM0 -b 115200

# Reset ESP32 (press RST button or unplug/replug USB)

# Terminal 2: Check topics
ros2 topic list
```

### Issue 5: dmesg Permission Denied

**Error:** `operation not permitted`
**Solution:** Use `sudo dmesg` or skip (not critical for setup)

---

# PART 10: Final Working Configuration

## System Information

- **OS:** Ubuntu 24.04 LTS
- **ROS2:** Humble Hawksbill
- **ESP32 Board:** ESP32 WROOM DevKit
- **Serial Port:** /dev/ttyACM0
- **Baud Rate:** 115200
- **Arduino IDE:** 2.3.2

## Environment Variables (Working Configuration)

bash

```
# DO NOT SET these - leave unset for compatibility
# unset RMW_IMPLEMENTATION
# unset ROS_DOMAIN_ID

# These should be set
source /opt/ros/humble/setup.bash
source ~/ros2_ws/install/setup.bash
```

## Working Commands

**Start micro-ROS Agent:**

bash

```
ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/ttyACM0 -b 115200
```

**List Topics:**

bash

```
ros2 topic list
```

**Echo Messages:**

bash

```
ros2 topic echo /esp32_chatter
```

**List Nodes:**

bash

```
ros2 node list
```

**Check Topic Info:**

bash

```
ros2 topic info /esp32_chatter
```

**Check Topic Frequency:**

bash

```
ros2 topic hz /esp32_chatter
```

## Installed Packages

**From APT:**

- ros-humble-rmw-cyclonedds-cpp
- python3-colcon-common-extensions
- python3-serial

**Built from Source:**

- micro_ros_setup (humble branch)
- micro_ros_agent (humble branch)

**Arduino Libraries:**

- micro_ros_arduino
- ESP32 board support (Espressif Systems)

## Group Memberships

```
jittu : jittu adm tty dialout cdrom sudo dip plugdev lpadmin lxd sambashare
```

**Critical groups for ESP32:**

- `dialout` ✅
- `tty` ✅

---

# Quick Start Reference (After Setup)

## Every Time You Want to Use ESP32 with ROS2:

### Terminal 1: Start Agent

bash

```bash
source /opt/ros/humble/setup.bash
source ~/ros2_ws/install/setup.bash
ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/ttyACM0 -b 115200
```

### Press RESET on ESP32 or replug USB

### Terminal 2: Check Communication

bash

```bash
source /opt/ros/humble/setup.bash
ros2 topic list
ros2 topic echo /esp32_chatter
```

---

# Common Errors and Quick Fixes

## "Serial port not found"

bash

```bash
# Check port exists
ls /dev/ttyACM*

# Check permissions
ls -l /dev/ttyACM0

# Temporary fix
sudo chmod 666 /dev/ttyACM0
```

### "Topics not showing"

bash

```bash
# Unset RMW
unset RMW_IMPLEMENTATION
unset ROS_DOMAIN_ID

# Restart agent and reset ESP32
```

### "Upload failed in Arduino"

- Hold BOOT button while clicking Upload
- Release when "Connecting..." appears
- Try different USB cable

### "Permission denied on port"

bash

```bash
# Check groups
groups $USER

# Should include dialout and tty
# If not, add and reboot
sudo usermod -a -G dialout $USER
sudo reboot
```

---

## Additional Resources

### Official Documentation

- micro-ROS: https://micro.ros.org/
- ROS2 Humble: https://docs.ros.org/en/humble/
- ESP32 Arduino: https://docs.espressif.com/projects/arduino-esp32/

### GitHub Repositories

- micro-ROS setup: https://github.com/micro-ROS/micro_ros_setup
- micro-ROS agent: https://github.com/micro-ROS/micro-ros-agent
- micro-ROS Arduino: https://github.com/micro-ROS/micro_ros_arduino

---

# Success Criteria Checklist

✅ Port `/dev/ttyACM0` detected
✅ User in `dialout` and `tty` groups
✅ ROS2 Humble sourced
✅ Arduino IDE installed with ESP32 support
✅ micro_ros_arduino library installed
✅ micro-ROS agent built from source
✅ Code uploaded to ESP32
✅ Agent connects to ESP32
✅ Topic `/esp32_chatter` visible
✅ Node `/esp32_node` visible
✅ Messages "Hello from ESP32" received

**Status: FULLY WORKING ✅✅✅**

---

# Next Steps

1. **Add Sensors:** Ultrasonic, IMU, temperature sensors
2. **Control Actuators:** Servo motors, DC motors
3. **WiFi Mode:** Use WiFi instead of serial connection
4. **Multiple Topics:** Publish multiple sensor data
5. **Subscribers:** Receive commands from ROS2
6. **Services:** Create request/response patterns
7. **Parameters:** Configure ESP32 via ROS2 parameters

---

**Document Version:** 1.0
**Date:** October 2025
**Status:** Complete and Tested
**Hardware:** ESP32 WROOM DevKit
**Software:** ROS2 Humble + Ubuntu 24.04

**END OF DOCUMENTATION**