

2. PENDAHULUAN

2.1 Latar Belakang

Dalam ekosistem rumah sakit modern (*Smart Hospital*), efisiensi dan kecepatan respons tenaga medis merupakan faktor vital dalam keselamatan pasien. Sistem pemanggilan perawat (*Nurse Call System*) konvensional umumnya masih berbasis analog (kabel) yang kaku, sulit dikembangkan, dan tidak terintegrasi dengan sistem data digital.

Seiring dengan perkembangan teknologi Robotika Medis, diperlukan sistem yang lebih cerdas, fleksibel, dan berbasis data. Oleh karena itu, proyek ini mengembangkan prototipe sistem *Nurse Call* nirkabel yang memanfaatkan Robot Operating System 2 (ROS 2). Penggunaan ROS 2 memungkinkan sistem ini dikembangkan lebih lanjut menjadi robot asisten otonom di masa depan.

2.2 Deskripsi Proyek

Proyek ini berjudul "Smart Nurse Call System berbasis Micro-ROS". Sistem ini adalah implementasi sistem cerdas berbasis ROS 2 yang menghubungkan perangkat keras mikrokontroler (ESP32) dengan komputer pusat (*Nurse Station*) melalui jaringan WiFi.

Sesuai dengan spesifikasi tugas UAS, sistem ini memenuhi kriteria berikut:

1. Arsitektur Node Terdistribusi: Terdapat minimal dua node yang berkomunikasi, yaitu Node Mikrokontroler (Client) dan Node Laptop (Server).
2. Mekanisme Publisher (Input): Menggunakan tombol fisik (*Push Button*) sebagai simulasi pasien memanggil bantuan. Data ini dikirim (*publish*) ke sistem pusat.
3. Mekanisme Subscriber (Output): Menggunakan LED dan Buzzer sebagai aktuator notifikasi. Perangkat ini menerima perintah (*subscribe*) dari sistem pusat untuk menyalakan alarm.

2.3 Tujuan Pengembangan

1. Mengimplementasikan protokol komunikasi Micro-ROS untuk menjembatani perangkat *low-level* (ESP32) dengan ekosistem ROS 2.
2. Membangun sistem kendali lingkar tertutup (*Closed-Loop System*) dimana input fisik diproses secara digital untuk menghasilkan output fisik secara *real-time*.
3. Memenuhi Capaian Pembelajaran Mata Kuliah (CPMK) dalam menganalisis tren perkembangan teknologi biomedis melalui penerapan IoT dan Robotika.

3. LANGKAH-LANGKAH PEMBUATAN SISTEM

Proses pembuatan sistem **Smart Nurse Call** ini dibagi menjadi 5 tahapan utama:

1. **Persiapan Lingkungan Software (ROS 2 & WSL)**

2. **Konfigurasi Jaringan & Keamanan**
3. **Perakitan Perangkat Keras (Hardware)**
4. **Pemrograman Firmware (ESP32)**
5. **Pemrograman Logic Node (Laptop)**

TAHAP 1: Instalasi Software (Environment Setup)

Sistem operasi utama yang digunakan adalah **Windows 11** dengan subsistem Linux (**WSL 2**) yang menjalankan **Ubuntu 24.04**.

1.1. Instalasi WSL 2 (Ubuntu 24.04)

1. Buka **PowerShell** sebagai Administrator.
2. Jalankan perintah instalasi:

```
wsl --install -d Ubuntu-24.04
```
3. Restart komputer jika diminta.
4. Buka aplikasi "Ubuntu" dari Start Menu, lalu buat username dan password UNIX baru.

1.2. Instalasi ROS 2 Jazzy Jalisco

Masuk ke terminal Ubuntu, lalu jalankan perintah berikut baris per baris:

1. **Set Locale:**

```
locale # Pastikan UTF-8  
sudo apt update && sudo apt install locales  
sudo locale-gen en_US.UTF-8  
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8  
export LANG=en_US.UTF-8
```

2. **Tambahkan Repository ROS 2:**

```
sudo apt install software-properties-common  
sudo add-apt-repository universe  
sudo apt update && sudo apt install curl -y  
sudo curl -sSL  
https://raw.githubusercontent.com/ros/rosdistro/master/ros.key  
-o /usr/share/keyrings/ros-archive-keyring.gpg  
echo "deb [arch=$(dpkg --print-architecture) signed-  
by=/usr/share/keyrings/ros-archive-keyring.gpg]  
http://packages.ros.org/ros2/ubuntu $(. /etc/os-release &&
```

```
echo $UBUNTU_CODENAME) main" | sudo tee  
/etc/apt/sources.list.d/ros2.list > /dev/null
```

3. **Install Paket ROS 2 (Desktop):**

```
sudo apt update  
sudo apt upgrade  
sudo apt install ros-jazzy-desktop
```

4. **Setup Environment:** Agar perintah ROS dikenali setiap terminal dibuka:

```
echo "source /opt/ros/jazzy/setup.bash" >> ~/.bashrc  
source ~/.bashrc
```

1.3. Instalasi Docker & Micro-ROS Agent

Micro-ROS Agent dijalankan menggunakan Docker agar lebih stabil.

1. **Install Docker di Windows:** Unduh dan install **Docker Desktop for Windows**.
2. **Pull Image Micro-ROS:** Buka terminal Ubuntu, tarik image versi Jazzy:

```
sudo docker pull microros/micro-ros-agent:jazzy
```

TAHAP 2: Konfigurasi Jaringan & Keamanan

Tahap ini krusial agar ESP32 bisa berkomunikasi dengan Laptop melalui WiFi.

2.1. Mengaktifkan WSL Mirrored Mode

Agar IP Ubuntu sama dengan IP Windows (sehingga terbaca ESP32).

1. Di Windows, buka **File Explorer**.
2. Ketik %UserProfile% di address bar.
3. Buat file baru bernama .wslconfig (pastikan tidak ada ekstensi .txt).
4. Isi file dengan konfigurasi berikut:

```
[wsl2]  
networkingMode=mirrored  
dnsTunneling=true  
firewall=true  
autoProxy=true
```

5. Restart WSL via PowerShell: wsl --shutdown.

2.2. Membuka Firewall Windows

Membuka Port 8888 (UDP) agar data ESP32 tidak diblokir.

1. Buka **PowerShell (Run as Administrator)**.

2. Jalankan perintah:

```
New-NetFirewallRule -DisplayName "MicroROS Agent" -Direction Inbound -LocalPort 8888 -Protocol UDP -Action Allow
```

TAHAP 3: Perakitan Hardware

Merangkai komponen elektronik pada Breadboard.

3.1 Alat dan Bahan (Komponen Sistem)

Sistem ini terdiri dari perangkat keras (*hardware*) di sisi pasien dan perangkat lunak (*software*) di sisi server.

A. Perangkat Keras (Hardware):

1. **ESP32 DOIT DEVKIT V1:** Mikrokontroler utama yang berfungsi sebagai pemroses data dan modul komunikasi WiFi.
2. **Push Button (Tactile Switch):** Sensor input manual untuk memicu sinyal darurat.
3. **LED Merah (5mm):** Indikator visual untuk status bahaya/panggilan aktif.
4. **LED Hijau (5mm):** Indikator visual untuk status aman/standby.
5. **Resistor 330 Ohm (2 buah):** Komponen penghambat arus listrik yang **wajib** dipasang pada kaki LED agar LED tidak kelebihan tegangan dan terbakar (putus).
6. **Active Buzzer (5V):** Aktuator suara untuk memberikan notifikasi alarm.
7. **Breadboard & Kabel Jumper:** Media perakitan rangkaian non-permanen.
8. **Kabel Micro USB:** Untuk catu daya dan unggah program.
9. **Laptop:** Sebagai *Nurse Station* (Server).

B. Perangkat Lunak (Software):

1. **Sistem Operasi:** Windows 11 (Host) & Ubuntu 24.04 (WSL).
2. **Robot Operating System:** ROS 2 Jazzy Jalisco.
3. **Middleware:** Micro-ROS Agent (via Docker).
4. **IDE:** Arduino IDE (v2.x) & VS Code (Optional).

3.2 Perancangan Perangkat Keras (Wiring)

Perakitan dilakukan dengan menghubungkan komponen ke GPIO ESP32. Sangat penting untuk memasang **Resistor 330 Ohm** secara seri dengan LED.

Komponen	Kaki Komponen	Pin ESP32	Keterangan Sambungan
Push Button	Kaki 1	GPIO 16 (RX2)	Mode INPUT_PULLUP
	Kaki 2	GND	-

LED Merah	Kaki Positif (Anoda)	GPIO 23	-
	Kaki Negatif (Katoda)	GND	Lewat Resistor 330Ω (Seri)
LED Hijau	Kaki Positif (Anoda)	GPIO 22	-
	Kaki Negatif (Katoda)	GND	Lewat Resistor 330Ω (Seri)
Buzzer	Kaki Positif (+)	GPIO 25	-
	Kaki Negatif (-)	GND	Langsung ke Ground

TAHAP 4: Pemrograman Firmware (ESP32)

4.1. Setup Arduino IDE

1. Install **Arduino IDE 2.0+**.

2. Install Board ESP32:

- File -> Preferences -> Additional Board Manager URLs:
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
- Tools -> Board -> Board Manager -> Cari "esp32" by Espressif Systems -> Install.

3. Install Library Micro-ROS:

- Download library .zip dari GitHub micro_ros_arduino.
- Sketch -> Include Library -> Add .ZIP Library.

4.2. Upload Kode Program

1. Salin kode lengkap (robotika_medis.ino) ke Arduino IDE.

2. **PENTING:** Edit bagian konfigurasi jaringan sesuai kondisi saat ini:

```
char ssid[] = "NamaWiFiAnda";
char password[] = "PasswordWiFiAnda";
char agent_ip[] = "192.168.x.x"; // Cek pakai 'ip a' di Ubuntu
```

3. Hubungkan ESP32 via USB.

4. Pilih Board: DOIT ESP32 DEVKIT V1.

5. Klik Upload.

TAHAP 5: Pengujian Komunikasi Manual (CLI)

Sebelum membuat program otomatis, pengujian manual dilakukan menggunakan *Command Line Interface* (CLI) ROS 2 untuk memastikan jalur data Input dan Output sudah terhubung dengan benar tanpa *bug*.

5.1. Persiapan Terminal

Buka terminal baru di Ubuntu (pastikan Agent Micro-ROS sudah berjalan di terminal lain).

Masuk ke lingkungan ROS 2:

```
sudo docker run -it --rm --net=host ros:jazzy
```

5.2. Pengujian Input (Monitoring Tombol)

Langkah ini bertujuan memverifikasi apakah data dari ESP32 masuk ke Laptop.

1. Jalankan perintah *echo* (mendengar):

```
ros2 topic echo /button_state --no-daemon
```

2. **Aksi:** Tekan dan tahan tombol fisik di ESP32.
3. **Hasil:** Terminal menampilkan data: true. Saat dilepas, menampilkan data: false.

5.3. Pengujian Output (Kendali Lampu)

Langkah ini bertujuan memverifikasi apakah Laptop bisa memerintah ESP32.

1. **Nyalakan Alarm (Simulasi Bahaya):**

```
ros2 topic pub --once /led_command std_msgs/msg/Bool "{data: true}"
```

Hasil: LED Merah menyala dan Buzzer berbunyi.

2. **Matikan Alarm (Simulasi Aman):**

```
ros2 topic pub --once /led_command std_msgs/msg/Bool "{data: false}"
```

Hasil: LED Merah mati dan LED Hijau menyala.

TAHAP 6: Implementasi Logic Node (Otomatisasi Sistem)

Setelah pengujian manual berhasil, tahap terakhir adalah membuat "Otak" (*Node*) menggunakan Python agar sistem dapat merespons secara otomatis tanpa perlu mengetik perintah manual.

1. Buka terminal Ubuntu dan buat file script Python:

```
nano nurse_bot.py
```

2. Masukkan kode program berikut:

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import Bool
```

```
class NurseBot(Node):
    def __init__(self):
        super().__init__('nurse_bot')
        # Publisher: Mengirim perintah ke Lampu/Buzzer
        self.publisher_ = self.create_publisher(Bool,
'/led_command', 10)
        # Subscriber: Menerima data dari Tombol
        self.subscription = self.create_subscription(Bool,
'/button_state', self.callback, 10)
        self.get_logger().info("SISTEM NURSE CALL AKTIF...
Menunggu Panggilan.")

    def callback(self, msg):
        # Logika Sederhana: Input = Output
        # Jika Tombol True (Ditekan) -> Kirim Perintah True
        (Nyala)
        cmd = Bool()
        cmd.data = msg.data
        self.publisher_.publish(cmd)

        # Tampilkan Log
        if msg.data:
            self.get_logger().warn("⚠ DARURAT! Pasien
Memanggil Bantuan!")
        else:
            self.get_logger().info("✅ AMAN. Tombol Dilepas.")

def main(args=None):
    rclpy.init(args=args)
    node = NurseBot()
    rclpy.spin(node) # Loop agar program tidak berhenti
    rclpy.shutdown()
```

```
if __name__ == '__main__':
    main()
```

3. Simpan file (Ctrl+X, Y, Enter).
4. Jalankan sistem otomatisasi:

```
python3 nurse_bot.py
```

Hasil: Sekarang saat tombol ditekan, alarm akan menyala secara otomatis.

TAHAP 7: Uji Coba Keseluruhan (Running System)

Berikut adalah urutan menyalakan sistem agar berhasil:

1. **Jalankan Agent (Di Terminal 1):**

```
sudo docker run -it --rm --net=host microros/micro-ros-agent:jazzy udp4
--port 8888 -v6
```

2. **Nyalakan ESP32:** Tekan tombol **RESET (EN)** di ESP32. Pastikan di Terminal 1 muncul: Session established.
3. Jalankan Otak/Bot (Di Terminal 2):

```
python3 nurse_bot.py
```

4. **Demonstrasi:**

- Tekan Tombol Fisik -> Lampu Merah Nyala & Terminal Laptop muncul peringatan "DARURAT".
- Lepas Tombol -> Lampu Hijau Nyala.

4. CATATAN KENDALA DAN PENYELESAIAN MASALAH

Selama proses penggerjaan proyek Sistem *Nurse Call* berbasis Micro-ROS ini, tim kami menghadapi beberapa tantangan teknis, mulai dari masalah kompilasi kode hingga konfigurasi jaringan yang kompleks. Berikut adalah rincian kendala yang terjadi beserta analisis penyebab dan solusi yang diterapkan:

No.	Jenis Kendala	Deskripsi & Penyebab Masalah	Solusi (Troubleshooting)
1.	Error Kompilasi Arduino	Pesan Error: invalid conversion from 'const char*' to 'char*'	Mengubah deklarasi variabel konfigurasi jaringan dari const char* menjadi char[].

		<p>Penyebab: Ketidakcocokan tipe data pada library micro_ros_arduino. Library meminta tipe data char* (array karakter yang bisa diubah), namun kami awalnya mendefinisikan SSID dan Password sebagai const char* (konstanta).</p>	<p><i>Contoh:</i> char ssid[] = "..."</p>
2.	Koneksi Gagal (Session)	<p>Gejala: ESP32 tidak dapat terhubung ke Agent (log "Session Established" tidak muncul).</p> <p>Penyebab: Isu pada jaringan WSL 2 (Windows Subsystem for Linux). IP Address default WSL (172.x.x.x) menggunakan sistem NAT, sehingga tidak terlihat oleh ESP32 yang berada di jaringan WiFi lokal (192.168.x.x).</p>	<p>Mengubah konfigurasi jaringan WSL menjadi Mode Mirrored. Kami membuat file konfigurasi .wslconfig agar IP Address Ubuntu menjadi sama persis dengan IP Address Windows, sehingga bisa diakses oleh ESP32.</p>
3.	Firewall Blocking	<p>Gejala: IP sudah benar, tetapi data tetap tidak masuk ke Agent.</p> <p>Penyebab: Windows Firewall secara <i>default</i> memblokir koneksi masuk (<i>Inbound</i>) pada port UDP 8888 yang digunakan oleh Micro-ROS.</p>	<p>Membuat aturan baru (<i>New Inbound Rule</i>) pada Windows Firewall menggunakan PowerShell untuk mengizinkan protokol UDP pada Port 8888.</p>

4.	ROS 2 Daemon Timeout	<p>Pesan Error: TimeoutError: [Errno 110] Connection timed out saat menjalankan perintah CLI ros2 topic echo.</p> <p>Penyebab: Servis latar belakang ROS 2 (<i>Daemon</i>) mengalami kegagalan komunikasi akibat perubahan konfigurasi jaringan yang mendadak atau isolasi container Docker.</p>	Menggunakan argumen --no-daemon pada setiap perintah CLI untuk memintas (<i>bypass</i>) penggunaan servis daemon, atau melakukan restart total pada container Docker.
5.	Missing Executable	<p>Pesan Error: ros2: command not found saat mencoba mengecek topik di dalam container Agent.</p> <p>Penyebab: Image Docker microros/micro-ros-agent bersifat minimalis dan tidak menyertakan paket <i>ros-core</i> lengkap.</p>	Menjalankan container terpisah menggunakan image ros:jazzy pada jaringan yang sama (--net=host) khusus untuk keperluan <i>debugging</i> dan pengiriman perintah manual.

4.2 Evaluasi Sistem

Meskipun terdapat kendala-kendala di atas, sistem akhirnya berhasil berjalan stabil setelah semua solusi diterapkan. Beberapa poin evaluasi dari penggeraan proyek ini adalah:

1. **Pentingnya Konfigurasi Jaringan:** Dalam proyek IoT yang melibatkan subsistem (seperti WSL) dan Docker, pemahaman tentang *IP Routing* dan *Firewall* sangat krusial. Mode jaringan host atau mirrored terbukti menjadi solusi paling efektif.
2. **Keandalan Micro-ROS:** Setelah koneksi terbentuk, protokol XRCE-DDS terbukti sangat andal. Data dari tombol dikirim secara *real-time* dengan latensi yang hampir tidak terasa, memenuhi standar kebutuhan sistem medis sederhana.

3. **Fleksibilitas ROS 2:** Penggunaan Node Python memudahkan pembuatan logika otomatisasi ("Otak Sistem") dibandingkan harus memprogram logika kompleks langsung di dalam mikrokontroler.