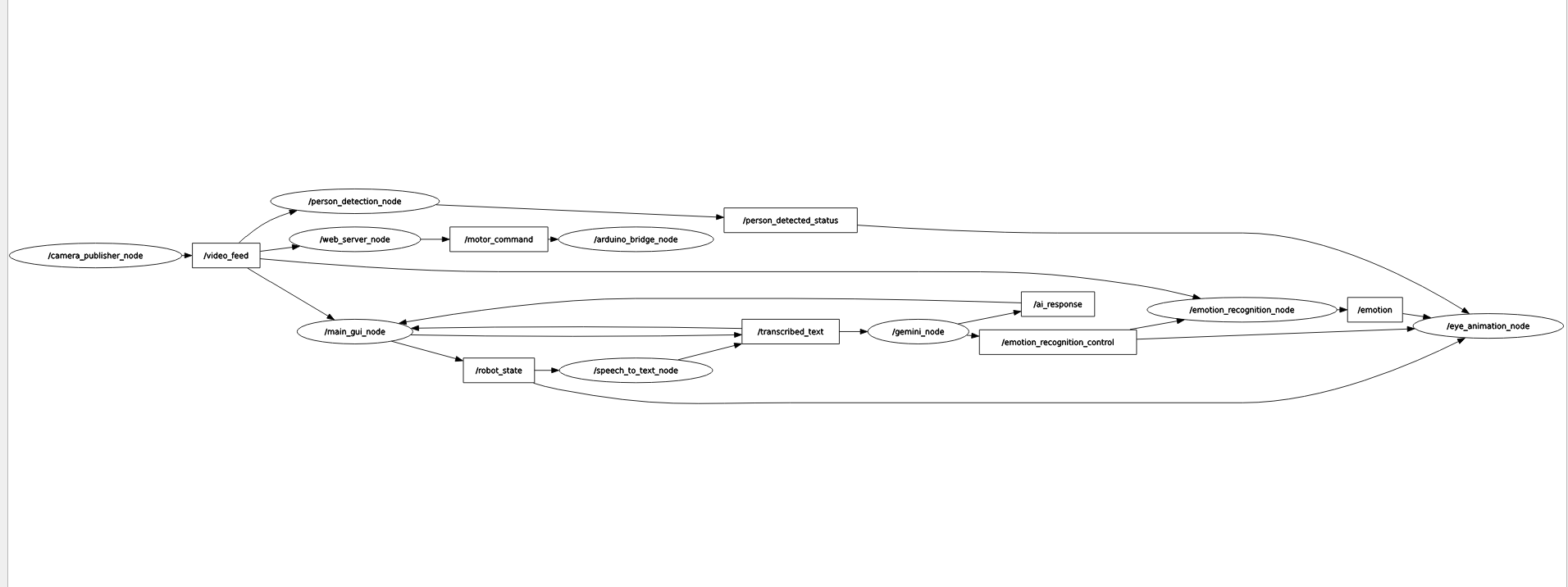
# TIST AI Robot Assistant - ROS 2 Project

This repository contains the complete ROS 2 workspace for an interactive AI robot assistant developed for the Toc H Institute of Science and Technology (TIST). The robot can see users, listen to their questions, provide intelligent answers using Google's Gemini AI, and express itself through an animated face. It can also be controlled remotely via a mobile application.



*The complete RQT graph showing the relationship between all nodes and topics.*

## 🤖 System Architecture

This project is built on a modular, multi-package ROS 2 architecture. Each package is responsible for a distinct part of the robot's functionality, communicating with other packages via a standardized set of ROS 2 topics.

* **input Package (The Senses)**: This package acts as the robot's sensory system. It contains nodes that interface directly with the hardware (camera, microphone) to capture real-world data and publish it to the ROS 2 network.
* **ai Package (The Brain)**: This is the central processing unit of the robot. It contains nodes for high-level logic, including person and emotion detection from the video feed, and a node to process user speech with the Gemini conversational AI.
* **ui Package (The Face & Voice)**: This package handles all human-robot interaction. It includes the main tkinter GUI that displays information to the user, a pygame node that renders the robot's expressive animated face on a dedicated screen, and the Text-to-Speech engine that gives the robot its voice.
* **remote\_control Package (The Nerves)**: This package provides the interface for manual remote control. It runs a web server to stream video to a mobile app and translates joystick commands from the app into motor commands for the robot's Arduino-based motor controller.

## ✨ Key Features

* ✅ **Person Detection**: Uses computer vision to detect when a person is in front of the robot, enabling it to initiate interactions.
* ✅ **Speech-to-Text**: Captures user speech and transcribes it to text using the Google Web Speech API.
* ✅ **Conversational AI**: Employs Google's Gemini AI to understand user queries, maintain conversational context, and generate intelligent, natural-sounding responses.
* ✅ **Emotion Recognition**: Can analyze a user's facial expression to determine their dominant emotion (e.g., happy, sad, angry) when instructed.
* ✅ **Expressive Animations**: Features a full-screen animated face that reacts to the state of the conversation, greets users, and can mirror the user's emotion.
* ✅ **Interruptible Text-to-Speech**: The AI's responses are spoken aloud using Google's TTS, but the user can interrupt at any time by speaking or typing.
* ✅ **Remote Control**: A web-based remote control system streams live video and accepts motor commands from a dedicated mobile app.
* ✅ **Modular ROS 2 Architecture**: Built with a professional, multi-package structure that separates concerns for robustness and scalability.

## 🌐 ROS 2 Topics and Communication

The packages communicate through the following ROS 2 topics:

| **Topic Name** | **Message Type** | **Publisher(s)** | **Subscriber(s)** | **Description** |
| --- | --- | --- | --- | --- |
| /video\_feed | sensor\_msgs/Image | input | ai (person/emotion), ui (main\_gui), remote\_control (web\_server) | The raw video stream from the camera. |
| /transcribed\_text | std\_msgs/String | input (stt), ui (gui) | ai (gemini\_node) | Contains the text transcribed from user speech or typed into the GUI. |
| /ai\_response | std\_msgs/String | ai (gemini\_node) | ui (main\_gui) | The final text response generated by the Gemini AI. |
| /robot\_state | std\_msgs/String | ui (main\_gui) | input (stt), ui (eye\_animation) | The robot's current state (e.g., "listening", "speaking") to manage actions. |
| /person\_detected\_status | std\_msgs/Bool | ai (person\_detection) | ui (eye\_animation) | True if a person is detected, False otherwise. |
| /emotion\_recognition\_control | std\_msgs/String | ai (gemini\_node) | ai (emotion\_recog), ui (eye\_animation) | Commands to "start\_mirroring" or "stop\_mirroring". |
| /emotion | std\_msgs/String | ai (emotion\_recog) | ui (eye\_animation) | The dominant emotion detected on the user's face (e.g., "happy", "sad"). |
| /motor\_command | std\_msgs/String | remote\_control (web) | remote\_control (arduino) | Motor commands (e.g., "forward", "stop") for the Arduino. |

## 🛠️ Setup and Installation

### Prerequisites

* Ubuntu 22.04 with ROS 2 Humble Hawksbill installed.
* A working webcam and microphone.
* An Arduino Uno (or similar) for motor control.

### 1. Clone the Repository

Create a ROS 2 workspace and clone this repository into its src folder.

mkdir -p ~/ros2\_ws/src  
cd ~/ros2\_ws/src  
git clone <your-repo-url> .

### 2. Install System Dependencies

Install the necessary libraries for TTS, GUI, and hardware communication.

sudo apt-get update  
sudo apt-get install python3-pip festival espeak mbrola mbrola-us1

### 3. Install Python Dependencies

Install all the required Python packages using pip. It is recommended to create a requirements.txt file in your workspace root for this.

# In ~/ros2\_ws  
pip install speech\_recognition google-generativeai deepface face\_recognition dlib opencv-python Pillow requests Flask Flask-Cors pyserial gTTS pygame

### 4. Build the Workspace

Use colcon to build all the packages.

cd ~/ros2\_ws  
colcon build --symlink-install

## ▶️ Running the Robot

To run the full system, you will need to open **four** separate terminals. In each terminal, first source your workspace:

cd ~/ros2\_ws  
source install/setup.bash

Then, run one of the following launch commands in each terminal:

* **Terminal 1 (Input):**  
  ros2 launch input input\_launch.py
* **Terminal 2 (AI Brain):**  
  ros2 launch ai ai\_launch.py
* **Terminal 3 (User Interfaces):**  
  ros2 launch ui ui\_launch.py
* **Terminal 4 (Remote Control):**  
  ros2 launch remote\_control remote\_control\_launch.py

Your robot should now be fully operational. The main GUI and the eye animation screen will appear, and it will be ready to interact with users and be controlled from your mobile app.

## 💡 Troubleshooting

* **Node Not Found / Package Not Found:** You likely forgot to source your workspace (source install/setup.bash) in the new terminal.
* **Python ModuleNotFoundError:** This usually means the file structure is incorrect. Ensure your Python nodes are inside a nested directory with the same name as your package (e.g., input/input/speech\_to\_text\_node.py).
* **Webcam Not Showing:** Check that your camera is connected and not being used by another application. You may need to change the camera index in input/input/camera\_publisher.py.
* **No Sound from Microphone:** The wrong microphone index might be selected. The speech\_to\_text\_node will print a list of available microphones on startup; adjust the MICROPHONE\_INDEX variable in the script if needed.
* **"Could not connect to Arduino" Error:** Ensure the Arduino is plugged in and you have the correct port name (e.g., /dev/ttyUSB0 or /dev/ttyACM0) set in remote\_control/remote\_control/arduino\_bridge\_node.py. You may also need to grant user permissions to the serial port.