

# Robot Explain Yourself

## Enhancing Human-Robot Communication with Large Language Models

Willem Adnet

June 6, 2025

### Abstract

This report presents the design and implementation of a system for enhancing human-robot interaction by enabling a robot to explain its decisions, particularly those related to low-level perception data, through natural language using a Large Language Model (LLM). The project explores current methodologies, builds an integration framework, fine-tunes an LLM, and validates the system through user evaluations.

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Problem statement . . . . .	3
1.2	Objectives . . . . .	3
<b>2</b>	<b>Literature review</b>	<b>3</b>
2.1	Technologies and methodologies in Human-Robot Communication . . . . .	3
2.2	Interpretability of low-level perception in robotics . . . . .	3
2.3	Challenges and opportunities for understandable robotics . . . . .	4
<b>3</b>	<b>Framework Design</b>	<b>4</b>
3.1	Architecture Overview . . . . .	4
3.2	Installation of the Project . . . . .	4
3.2.1	Step-by-Step Installation Process . . . . .	5
3.3	System execution . . . . .	5
3.3.1	Running the Application . . . . .	5
3.3.2	Testing Framework . . . . .	6
3.4	Data flow and processing pipeline . . . . .	6
3.5	Project structure and organization . . . . .	6
3.6	Usage workflow . . . . .	6
3.7	Configuration and customization . . . . .	6
3.8	Technical requirements . . . . .	7
3.9	Troubleshooting and common issues . . . . .	7
3.10	Integration Capabilities . . . . .	7
<b>4</b>	<b>LLM Customization</b>	<b>7</b>
4.1	Model selection and comparison . . . . .	7
4.2	Prompt engineering . . . . .	7
4.3	Fine-tuning Approach . . . . .	7

<b>5</b>	<b>Human-Robot Interaction Prototype</b>	<b>8</b>
5.1	System implementation . . . . .	8
5.2	Core features . . . . .	8
5.3	Technical architecture . . . . .	8
5.4	Usage scenarios . . . . .	8
<b>6</b>	<b>Evaluation</b>	<b>9</b>
6.1	Evaluation methodology . . . . .	9
6.2	Quantitative results . . . . .	9
6.3	Qualitative findings . . . . .	9
6.4	Limitations and challenges . . . . .	9
<b>7</b>	<b>Conclusion</b>	<b>9</b>
7.1	Summary of contributions . . . . .	9
7.2	Impact and implications . . . . .	10
7.3	Future Work . . . . .	10
7.4	Final Remarks . . . . .	10

# 1 Introduction

The integration of artificial intelligence with robotics has opened new frontiers in human-robot interaction (HRI). As robots become increasingly autonomous and deployed in complex real-world environments, the need for transparent and interpretable decision-making becomes paramount. This project addresses the challenge of making robotic systems more explainable by leveraging Large Language Models (LLMs) to translate low-level sensor data and perception information into natural language explanations.

## 1.1 Problem statement

Modern robots operate using complex algorithms that process vast amounts of sensor data to make navigation and behavioral decisions. However, these decisions often remain opaque to human users, creating a barrier to trust and effective collaboration. The challenge lies in bridging the gap between machine perception and human understanding.

The primary goal is to develop an AI-powered robot capable of evaluating its past decisions, particularly when revisiting locations. For example, the robot should be able to explain: "I recognize this area—I previously visited it on [date/time] and made a certain decision." If the environment has changed since the last visit, the robot should update its reasoning to reflect the new context, rather than relying solely on prior experiences.

## 1.2 Objectives

This project aims to:

- Design a framework for integrating LLMs with robotic perception systems
- Develop a prototype system that can explain robot path decisions in natural language
- Evaluate the effectiveness of LLM-generated explanations in enhancing human understanding
- Assess the impact on user trust and satisfaction in human-robot interactions

# 2 Literature review

## 2.1 Technologies and methodologies in Human-Robot Communication

**Dialogue management in HRI** The current state of dialogue management in human-robot interaction is reviewed by Lopes et al. [3]. They evaluate capabilities, methods, and challenges, and emphasize the need for structured approaches that effectively combine HRI with dialogue systems.

**Human-robot interaction** An overview of human-robot interaction is provided in a community-maintained source [6]. It emphasizes intuitive communication through speech, gestures, and facial expressions, and underlines the importance of designing robots that “feel” human.

## 2.2 Interpretability of low-level perception in robotics

**Explainable robotic systems in scenarios** Puiutta and Veith [1] discuss explainability in robotic reinforcement learning agents. They present methods for making decision probabilities interpretable to end-users.

**Subsumption architecture** The subsumption architecture [7] is an early robotics design that focuses on real-time responsiveness and intelligent behavior emerging from layered, embodied interactions.

## 2.3 Challenges and opportunities for understandable robotics

**Trust in explainable robots** Trust calibration and explanation-specificity in robots are discussed in Wang et al. [5], who offer insight into designing systems that feel transparent and reliable.

**Effects of robot explanations** The impact of robot explanations on user perception is further evaluated by the same authors [4]. Their findings suggest explanations make robots seem livelier and more human-like.

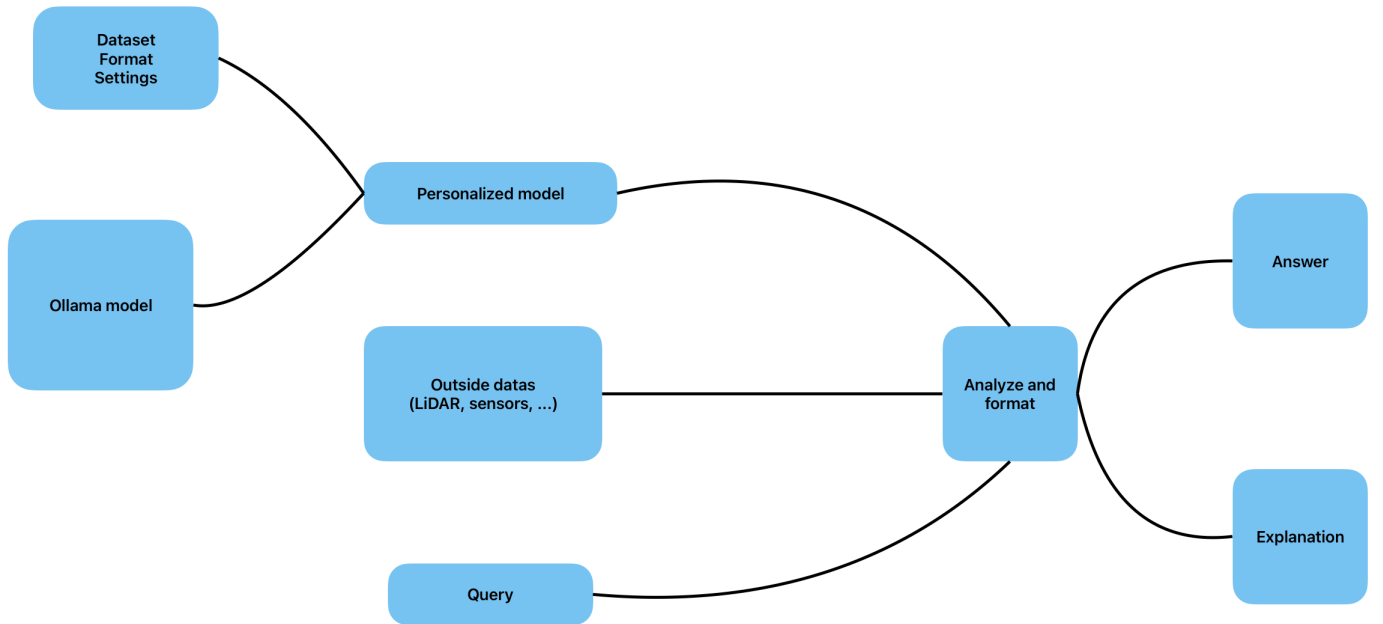
**Ethical black box** Professor Marina Jirotko’s concept of the “ethical black box” [2] suggests embedding inflight recorders in robots to promote accountability and auditability of robotic decisions.

## 3 Framework Design

### 3.1 Architecture Overview

The proposed framework integrates three main components:

1. **Perception Processing Layer:** Handles sensor data aggregation and context extraction
2. **LLM Integration Layer:** Manages prompt generation and response processing
3. **Human Interface Layer:** Provides natural language interaction capabilities



### 3.2 Installation of the Project

The project is designed to be easily deployable on any computer with minimal setup requirements. The following prerequisites are needed:

- **Python 3.8+** (recommended: Python 3.11)
- **Ollama** - Local LLM server for model hosting
- **llama3.2** model or compatible LLM model
- **make** (optional, for automation)
- **git** for repository cloning

### 3.2.1 Step-by-Step Installation Process

#### 1. Repository cloning

The project can be obtained from the GitHub repository:

```
git clone https://github.com/Vlor999/HCI.git
cd HCI
```

#### 2. Ollama installation and setup

Ollama serves as the local LLM server, providing the computational backend for natural language processing:

- Download and install Ollama from <https://ollama.com/download>
- Start the Ollama server: `ollama serve`
- Pull the required model: `ollama pull llama3.2` (or any other model)

#### 3. Project environment setup

The project uses automated setup through make commands:

```
make init
make install
```

This process creates the necessary directory structure and installs Python dependencies in a virtual environment.

#### 4. Optional code formatting

For development consistency:

```
make format
```

But if you also want to stay consistent with the all code you have to run the following command :

```
pre-commit install
```

Like that for the next commit that you'll do everything will be checked before you submit it.

### 3.3 System execution

#### 3.3.1 Running the Application

The system can be launched using either automated make commands or direct Python execution:

```
make run
```

or manually:

```
.venv/bin/python main.py
```

Obviously you can add some CLI commands and if you are into the current env you can simply run :

```
python main.py
```

If you want to know all the commands that can be used try the help CLI argument.

### 3.3.2 Testing Framework

The project includes comprehensive testing capabilities:

```
make test
```

For coverage analysis with HTML reporting:

```
make coverage
```

The coverage report can be viewed by opening `htmlcov/index.html` in a web browser.

## 3.4 Data flow and processing pipeline

The system processes robot perception data through a structured pipeline:

- Environmental/Weather context extraction from sensor readings
- Path analysis and decision point identification
- Prompt template generation with structured information
- LLM query processing and response generation
- Natural language explanation delivery to users

## 3.5 Project structure and organization

The framework follows a modular architecture with clear separation of concerns:

```
src/           # Source code modules
tests/         # Unit tests and test data
data/          # Example path scenarios (JSON)
log/           # Conversation logs (Markdown)
doc/           # Documentation and roadmap
evaluation/    # Evaluation scripts and results
Makefile       # Automation commands
requirements.txt # Python dependencies
```

## 3.6 Usage workflow

The typical user interaction follows this pattern:

1. The system displays current robot path and environmental context
2. Users can ask multiple questions about path decisions and conditions
3. The LLM processes queries and generates contextual explanations
4. Sessions are terminated with `exit` or `quit` commands
5. Conversation logs are automatically saved in Markdown format in the `log/` directory

## 3.7 Configuration and customization

The system supports various customization options:

- **Path scenarios:** Edit `data/paths.json` to add or modify navigation scenarios
- **LLM models:** The model name can be changed in the source code for different LLM variants
- **Logging:** Conversation logs are automatically generated and stored for analysis

### 3.8 Technical requirements

The framework is designed with several technical requirements in mind. It must support real-time processing to enable interactive use and immediate responses to user queries. The architecture should remain modular to facilitate adaptation across different robot platforms and allow for future extensions. Scalability is also essential, ensuring that the system can handle a variety of explanation types and increasing complexity as needed.

### 3.9 Troubleshooting and common issues

When deploying the system, several common issues may arise. First, it is important to ensure that the Ollama server is running by verifying that the `ollama serve` command is active and that the required model has been properly pulled. Port conflicts can also occur, as only one Ollama server instance should be running on port 11434 at any given time. Additionally, the Python environment must be correctly set up, make sure that the virtual environment (`.venv`) is properly activated before running any Python scripts. Finally, confirm that the necessary large language model (LLM) is downloaded and accessible to avoid runtime errors related to model availability.

### 3.10 Integration Capabilities

## 4 LLM Customization

### 4.1 Model selection and comparison

The project evaluated multiple LLM architectures:

Model	Size	Performance
Llama 3.2	2.0 GB	Baseline performance
nous-hermes2:latest	6.1 GB	small model that explain lightly
deepseek	8.1 GB	Good reasoning, efficient and clear
qwen3:30b-a3b	18 GB	big model that provides good answers but take some time

Table 1: LLM model comparison

### 4.2 Prompt engineering

Effective prompt design proved crucial for generating relevant explanations. The system uses structured prompts that include:

- Environmental context and sensor readings
- Historical path information
- User questions and interaction history
- Domain-specific constraints and objectives
- Additional datas provided by the human or captors

### 4.3 Fine-tuning Approach

The customization process involved:

1. Dataset creation with robot-specific scenarios
2. Prompt template

3. Response quality evaluation and iteration
4. Integration with robot perception systems (In the future)

## 5 Human-Robot Interaction Prototype

### 5.1 System implementation

The prototype system was implemented using Python with the following key components:

- **Path processing module:** Handles environmental data and context extraction
- **LLM interface:** Manages communication with local Ollama server
- **Conversation manager:** Maintains interaction history and context
- **User interface:** Provides command-line and potential interaction

### 5.2 Core features

The implemented system supports:

- Interactive questioning about robot path decisions
- Real-time explanation generation
- Context-aware responses based on environmental conditions
- Conversation logging and history management
- Multiple scenario support for testing and evaluation

### 5.3 Technical architecture

The system architecture follows a modular design:

- `robotPathExplanation.py`: Main application logic
- `path.py`: Data structures for path and environmental information
- `llmModel.py`: LLM integration and prompt management
- `conversationLogger.py`: Interaction recording and analysis

### 5.4 Usage scenarios

The prototype supports various interaction scenarios:

1. Path selection queries (e.g., "Which path should I take if I want the easiest route?")
2. Safety-related questions (e.g., "I have a heavy load. Which path is safest?")
3. Time-constrained decisions (e.g., "I am in a hurry but want to avoid danger")
4. Real-time updated decision (e.g., "Which path should i took according to the new datas that I provided ?")



## 6 Evaluation

### 6.1 Evaluation methodology

The system evaluation employed multiple approaches:

- **Automated testing:** Unit tests for core functionality and edge cases. Run when you commit something
- **Explanation quality assessment:** Keyword matching and semantic similarity analysis
- **User study design:** Interactive scenarios with human participants
- **Performance metrics:** Response time, accuracy, and user satisfaction measures (not really efficient but look if the answer makes sense)

### 6.2 Quantitative results

The evaluation framework assessed explanations based on:

- Keyword coverage score (target:  $\geq 75\%$ )
- Factual accuracy in decision reasoning
- Consistency across similar scenarios

### 6.3 Qualitative findings

Key observations from the evaluation include:

- Users appreciated natural language explanations
- Context-aware responses significantly improved user understanding
- Explanation quality varied with environmental complexity
- Interactive questioning enhanced user engagement and trust

### 6.4 Limitations and challenges

Identified limitations include:

- Computational overhead for real-time processing
- Occasional over-fitting in response patterns
- Context window limitations for extended conversations
- Need for domain-specific fine-tuning for optimal performance

## 7 Conclusion

### 7.1 Summary of contributions

This project successfully demonstrated the feasibility of using Large Language Models to enhance human-robot communication through natural language explanations of robotic decisions. Key contributions include:

- A novel integration framework for LLMs in robotic explanation systems

- A working prototype that translates low-level perception data into natural language
- Empirical evaluation demonstrating improved user understanding and engagement
- Open-source implementation available for further research and development

## 7.2 Impact and implications

The work addresses a critical gap in human-robot interaction by making robotic decision-making more transparent and interpretable. This has implications for:

- Increased user trust in autonomous systems
- Enhanced collaboration in human-robot teams
- Improved debugging and system maintenance
- Better user training and system adoption

## 7.3 Future Work

Several directions for future research emerge from this work:

- **Real-time integration:** Developing more efficient processing pipelines for live robot systems
- **Multimodal explanations:** Incorporating visual and gestural explanation modalities
- **Personalized explanations:** Adapting explanation style to individual user preferences
- **Domain expansion:** Extending beyond path planning to other robotic decision domains
- **Fine-tuning optimization:** Developing robot-specific language models for improved performance

## 7.4 Final Remarks

The integration of Large Language Models with robotic systems represents a promising approach to bridging the communication gap between humans and machines. As LLM technology continues to advance, we can expect even more sophisticated and natural human-robot interactions, ultimately leading to more effective and trustworthy autonomous systems.

The open-source nature of this implementation encourages further research and development in this important area of human-robot interaction.

## References

- [1] F. Cruz, R. Dazeley, P. Vamplew, and I. Moreira. Explainable robotic systems in reinforcement learning scenarios. 2020.
- [2] M. Jirotko. Ethical black box for robots, 2024. Wikipedia entry on Marina Jirotko’s work on accountability in robotics.
- [3] M. M. Reimann, F. A. Kunneman, C. Oertel, and K. V. Hindriks. A survey of dialogue management in human-robot interaction. 2023.
- [4] L. Sanneman and J. A. Shah. Effects of robot explanations on human perceptions. 2020.
- [5] Daisy Zhe Wang et al. Trust calibration in explainable robots. 2020.
- [6] Wikipedia contributors. Human–robot interaction, 2024. Wikipedia article.
- [7] Wikipedia contributors. Subsumption architecture, 2024. Wikipedia article.