ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Visual Chatting and Real-Time Acting Robot

Team #37

HAOZHE CHI
(haozhe4@illinois.edu)
MINGHUA YANG
(minghua3@illinois.edu)
JIATONG LI
(jl180@illinois.edu)
ZONGHAI JING
(zonghai2@illinois.edu)

TA: Enxin Song

May 10, 2024

Contents

1 Introduction

1.1 Problem Statement

Blind individuals often face significant difficulties when navigating unfamiliar environments, such as finding water dispensers in large public spaces. Additionally, there is a risk of injury from interacting with devices that dispense hot water. The emergence of large language models (LLMs) and large visual language models (LVLMs) offers a promising avenue for developing innovative solutions to these challenges.

1.2 Solution Overview

Our team develop an AI-enhanced robotic service system aimed at assisting blind individuals in navigating large public spaces to safely access and interact with water dispensers. This initiative addresses the significant challenges that visually impaired people face, such as the risk of injury from devices dispensing hot water and difficulty in locating such amenities.

The proposed system combines advanced technological components including large language models (LLMs) and large visual language models (LVLMs). These models process both visual inputs from a camera mounted on the user's head and verbal commands via speech-to-text AI technology. It provides real-time, actionable guidance and safety instructions. To enhance interaction experience between users and robot arm, we also develop a Raspberry Pi Auxiliary system that can provide auditory guidance through visual monitoring. Our system comprises the following key components:

- **Real-Time Visual and Verbal Input Processing:** A combination of a head-mounted camera and speech-to-text AI captures and analyzes the user's surroundings and voice commands.
- **Dynamic Guidance and Interaction:** The BLIP-2 model will provide navigation assistance, warn of potential dangers, and instruct on interacting with a water dispenser.
- **Autonomous Assistance:** A Universal Robot Arm UR3e, controlled by the Robot Operating System and instructed by the Raspberry Pi Auxiliary System, will autonomously refill the user's water bottle.
- **User Communication:** Audio feedback and instructions will be delivered through a Bluetooth headset, ensuring clear and effective communication.

Operational Process When a blind individual approaches a water dispenser, the system triggers a specific sequence of actions:

1. The Vision Language AI model guides the user to the water dispenser.
2. The Raspberry Pi Auxiliary System will then provide audio instructions to help user place their bottle in a designated location.

3. Subsequently, a robot arm, following instructions from the Raspberry Pi system, securely grasps the bottle, fills it with water from the dispenser, and then returns the filled bottle to the user.

1.3 Visual Aid

The visual illustration of our AI-enhanced robotic service system is shown in Figure 1.

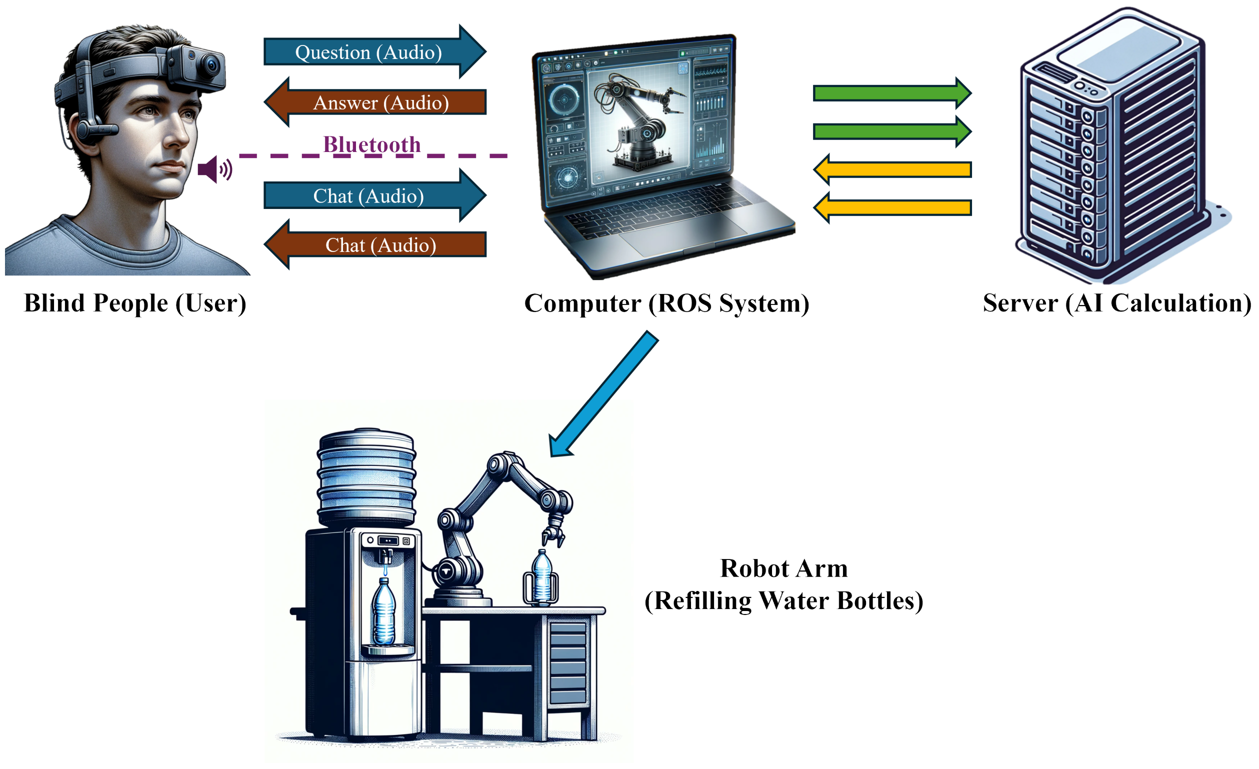


Figure 1: Visual Illustration of the AI-enhanced Robotic Service System

2 Design

2.1 Block Diagram

The overall block diagram of our AI-enhanced robotic service system is shown in Figure 2.

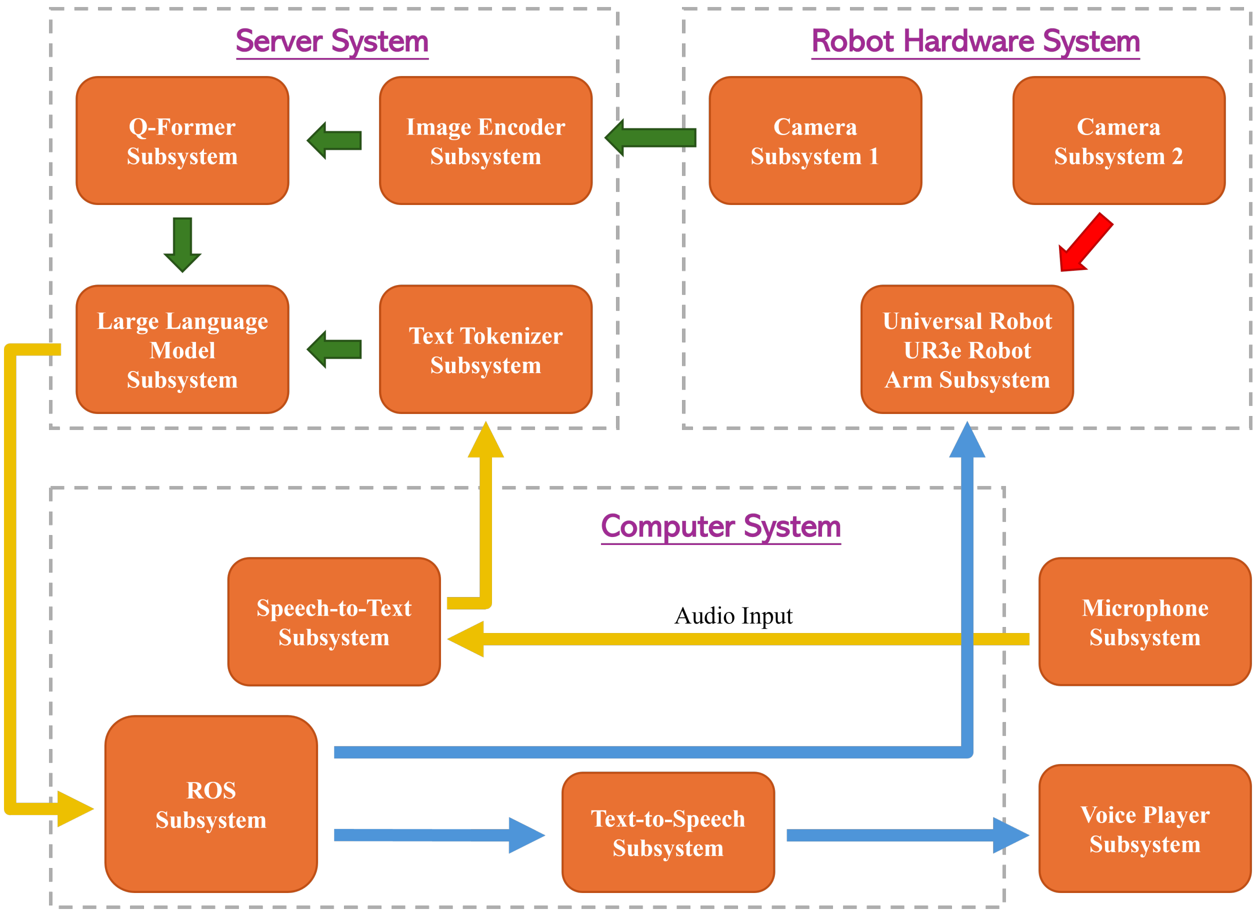


Figure 2: Block diagram of the AI-enhanced robotic service system (Green arrow: visual flow; Yellow arrow: text flow; Red arrow: attach itself to; Blue arrow: instruction flow), in which Camera Subsystem 1 is for blind people and Camera Subsystem 2 is for robot arm.

2.2 Subsystems Overview

2.2.1 Camera Subsystem

The Camera Subsystem is a pivotal element in our robotic framework, acting as the primary data collection point. Utilizing an iPhone camera mounted on the user's head, this subsystem captures the user's environment and streams video in real-time to a connected Mac. This setup ensures a continuous flow of high-resolution visual data to the Image Encoder Subsystem. The system's contribution is quantified by its ability to deliver high-resolution video under varying lighting conditions and maintain a seamless frame rate essential for subsequent processing stages. The interface with the Image Encoder Subsystem is defined by the video resolution, frame rate, and the real-time data transfer rate necessary for effective processing.

2.2.2 Image Encoder Subsystem

The Image Encoder Subsystem is an integral part of the robotics framework, responsible for converting visual input into a format suitable for advanced analysis. Leveraging the Vision Transformer (ViT) structure, specifically a **ViT-L/14** model, this subsystem provides a streamlined and feature-enriched representation of images captured by the Camera subsystem. It processes images through 32 queries of 768 dimensions each, aligning with the Q-Former's specifications for efficient interfacing. The subsystem's output, a compressed 32×768 matrix denoted as Z, presents a more efficient alternative to the initial 257×1024 ViT-L/14 image features. By reducing data dimensionality, it plays a crucial role in optimizing the computational workflow and facilitating swift data exchange with the Q-Former subsystem.

2.2.3 Q-Former Subsystem

The Q-Former Subsystem serves as a crucial component in the processing pipeline of our design, utilizing transformer architecture to elevate visual data into abstract representations. This subsystem ingeniously employs attention mechanisms to enhance the visual features received from the Image Encoder subsystem before passing them on to the Large Language Model (LLM) subsystem. Quantitatively, it boasts two transformer submodules that share self-attention layers to refine features from varying image resolutions. The image transformer submodule is tasked with the visual aspect, and the text transformer handles the encoding and decoding of textual information. Through this setup, the Q-Former ensures that the interaction between visual and textual data is not only seamless but also optimized for the highest efficiency in real-time processing.

2.2.4 Large Language Model Subsystem

The Large Language Model (LLM) Subsystem is a sophisticated computational unit within our robotics architecture, integral for synthesizing both visual and textual data into actionable text outputs. It processes embeddings from the Image Encoder and Text Tokenizer Subsystems using an advanced **Llama** model, a choice inspired by the **BLIP-2** [1] architecture which ensures comprehensive and nuanced text generation. This subsystem's outputs are specifically formatted to instruct the Robot Operating System (ROS) for executing tasks or providing responses. The quantitative measure of this subsystem's performance is assessed by the quality and relevance of text outputs generated, as well as the speed and accuracy with which it processes input embeddings into these outputs. The interface with the Image Encoder and Text Tokenizer is marked by the standardized embedding vectors received, while its output interface with the ROS subsystem is quantified by the command strings dispatched for robotic control.

2.2.5 Text Tokenizer Subsystem

The Text Tokenizer Subsystem is a pivotal component tasked with converting raw textual inputs into structured embeddings. It serves as an intermediary, translating spoken language captured by the Speech-to-Text subsystem into a format amenable to computational analysis. Utilizing a **BERT** model tokenizer ensures compatibility with advanced Large Language Models like Llama, enabling robust text interpretation. This subsystem significantly contributes to the overall design by ensuring that linguistic information is accurately represented and processed, facilitating the system's ability to comprehend and act upon user commands. The interface with the Speech-to-Text subsystem is defined by the text input stream, while the output interface with the Large Language Model consists of tokenized text embeddings.

2.2.6 Speech-to-Text Subsystem

The Speech-to-Text Subsystem is a key interface that translates auditory information into a digital text format, bridging human interaction and machine processing. Using an advanced open-source model, this subsystem decodes spoken language with high accuracy and low latency, making it an essential component for real-time applications. It quantitatively contributes to the overall design by providing accurate text conversion, serving as the initial processing step for voice commands. The efficacy of this subsystem is measured by its transcription accuracy and speed, which directly impacts the performance of the downstream Text Tokenizer subsystem.

2.2.7 Microphone Subsystem

The Microphone Subsystem is an integral component of our robotic system, tasked with capturing audio input from users in a clear and reliable manner. Utilizing a Bluetooth microphone, this subsystem offers flexibility and enhances the robot's ability to interact with its environment by ensuring high-quality audio capture. This audio is then transmitted to the Speech-to-Text subsystem, where it is converted into textual data for further processing. The performance of this subsystem is quantitatively measured by its audio capture fidelity, noise reduction capability, and the latency in transmitting the captured audio to the Speech-to-Text subsystem. Its seamless integration and reliability are critical for the effective operation of the robot's interactive capabilities.

2.2.8 ROS Subsystem

The ROS (Robot Operating System) Subsystem acts as the central control unit within our robotics framework, crucial for interfacing with both software components and hardware mechanisms. Leveraging MQTT, a lightweight messaging protocol, it facilitates real-time communication with the Raspberry Pi Auxiliary System, which manages the operational commands for the robot arm. This shift enhances the system's responsiveness and reliability, especially in low-bandwidth environments. The ROS Subsystem translates these commands into precise physical actions, coordinating closely with the Raspberry Pi to ensure seamless execution. Additionally, it continues to relay necessary responses back to the Text-to-Speech subsystem for user interaction. The system's performance is quantitatively assessed by its command execution latency, reliability in task execution, and the efficiency of inter-process communication.

2.2.9 Text-to-Speech Subsystem

The Text-to-Speech Subsystem is an essential communicative bridge in our robotics architecture, enabling the robot to convert textual responses into spoken words, thus facilitating a natural interaction with users. Utilizing the ***pyttsx3***[1], a versatile and open-source text-to-speech Python library, this subsystem translates textual data received from the ROS subsystem into audible speech, which is then relayed through the Voice Player subsystem for output. The choice of pyttsx3 not only supports a broad range of voices and languages but also ensures functionality without the need for internet connectivity. This subsystem's contribution to the overall design is quantitatively marked by its speech synthesis speed, clarity of the generated audio, and the seamless interface with the ROS and Voice Player subsystems, enabling the robot to provide timely and intelligible responses to user inquiries.

2.2.10 Voice Player Subsystem

The Voice Player Subsystem is a critical component for enabling the robot to audibly communicate with users, functioning as the final step in the interactive feedback loop. It takes the audio files generated by the Text-to-Speech Subsystem and plays them through a Bluetooth headset, ensuring clear and understandable speech output. This subsystem is essential for the robot's ability to provide audible responses to user queries or commands, enhancing the overall user experience. Its contribution to the design is quantified by its audio output clarity, playback latency, and compatibility with the Bluetooth headset, facilitating effective human-robot interaction.

2.2.11 Universal Robot UR3e Robot Arm Subsystem

The Universal Robot UR3e Robot Arm Subsystem, enhanced with a Makeblock Robot Gripper, is a critical component of our robotics architecture, providing high precision and flexibility for physical tasks. The gripper allows the arm to grasp and handle objects like water bottles more effectively. The gripper is controlled by a Raspberry Pi, which interfaces directly with the ROS subsystem to receive and execute detailed instructions. Equipped with six rotational joints, the subsystem executes movements with high precision, crucial for the accurate positioning and handling of objects within its operational environment. This subsystem's performance metrics include its reach, payload, repeatability, and the added functionality of the gripper's grasping capabilities.

2.2.12 Raspberry Pi Auxiliary Subsystem

The Raspberry Pi Auxiliary Subsystem serves as a sophisticated monitoring and interaction enhancer between the user and the robotic system, utilizing three cameras and a speaker for precise control and feedback. The first two cameras are dedicated to monitoring the water bottle on the desk, ensuring it is within the Robot Arm's reach and providing precise audio instructions for adjustments along the x-axis and y-axis, such

[1]https://pyttsx3.readthedocs.io/en/latest/

as "Move your bottle right/left/forward/back a bit." The third camera checks the positioning of the bottle at the water dispenser to guarantee accurate filling. This subsystem leverages advanced object recognition technology to pinpoint the bottle's location and provide verbal guidance accordingly. Connected to the PCB Water Dispenser Subsystem, the Robot Gripper, and the ROS Subsystem, it orchestrates a seamless interaction flow and enhances operational efficiency. This integration is critical for the system's functionality, offering real-time and intuitive user guidance.

2.2.13 PCB Water Dispenser Subsystem

The PCB Water Dispenser Subsystem is integral to the functionality and user interface of our robotic system, acting as a visual communicator for the operational status of the water dispensing process. By employing a light control mechanism with programmable LEDs, this subsystem indicates when the water is being dispensed (green light) and when the process is complete (red light), based on the input from the Raspberry Pi Auxiliary System. This direct, visual feedback mechanism is crucial for coordinating the actions of the robot arm, especially in guiding it to retrieve and return the filled water bottle to the user. The subsystem's design is quantitatively defined by the accuracy of signal reception, the precision of the internal timer for light transitions, and the reliability of sending completion signals back to the Raspberry Pi Auxiliary System. This ensures seamless integration within the broader system, enhancing the robot's interactive capabilities.

3 Design Details

Our AI-enhanced robotic service system has three main components, the Navigation System, the Raspberry Pi Auxiliary System, and the PCB Water Dispenser System.

3.1 Navigation System

The Navigation System includes the following part of work:

1. Deploy the speech-to-text and text-to-speech modules. This is essential for the AI model to process blind people's vocal input and give corresponding guidance.
2. Deploy the depth-map generation module and complete scripts for navigation algorithm. Since we do not use depth camera, we need a depth-map generation module to generate depth map from original visual image. Then the navigation algorithm would use this depth map to detect whether there is any potential danger.
3. Establish stable connection between my personal computer and the AI server, and realize efficient data transmission. This connection is important for stable real-time chatting.
4. Deploy Large Visual Language Models on the AI server. This is the core of our design, as the LVLM provides real-time guidance and instructions for blind people.

5. Apply model acceleration technique to the AI model. This is also important for reducing the delay of processing data with AI model.

3.1.1 Speech-to-Text and Text-to-Speech Modules

We deploy both speech-to-text and text-to-speech modules on PC (Mac OS). Specifically, we use a virtual machine VMware to install the Ubuntu system and deploy open-source modules on it. To realize the speech-to-text process, I use the pre-trained ***silero***[2] model.

Firstly we use the pyaudio library in Python to record the voice and save it as a wave file. Then we use the pre-trained silero model to process the wave file and turn it into text. To realize the text-to-speech process, we use the open-source project ***pyttsx3***. After receiving the JSON file from the AI server, our program would read the answer message from the JSON file and use pyttsx3 to turn it into audio and play it through Bluetooth devices. The detailed code is shown in Appendix A.

3.1.2 Navigation Algorithm

The navigation algorithm has different priorities as shown in Figure 3. For the highest priority, the system should remind the potential danger ahead immediately. The potential danger is detected through the depth map analysis. If there is no potential danger, then it comes to the second priority. For the second priority, the system should respond to blind people's vocal inputs. If there is also no vocal input from blind people, then it comes to the third priority. For the third priority, the system should repeat the navigation route that blind people should follow.

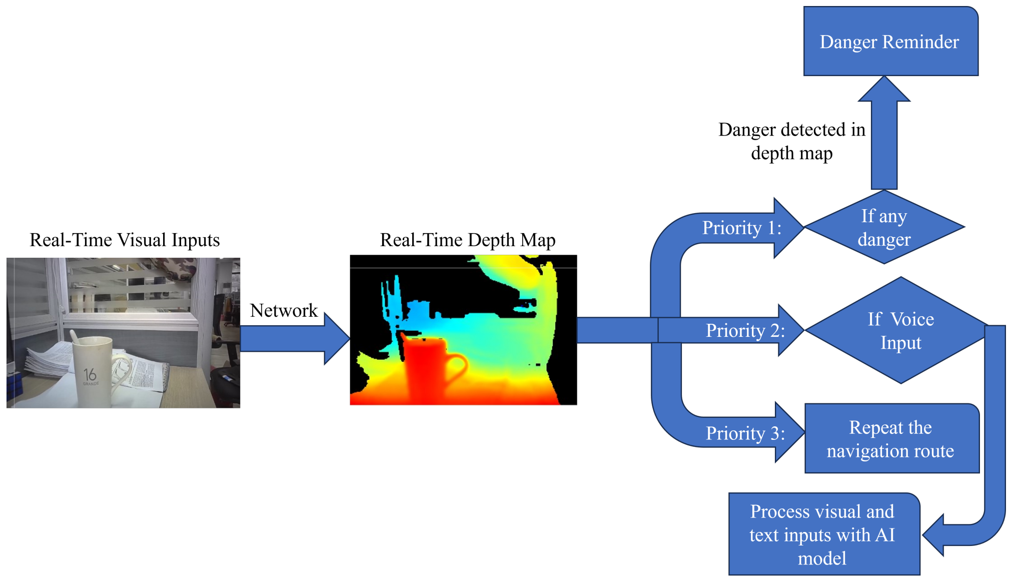


Figure 3: Navigation Algorithm Illustration

[2]https://github.com/snakers4/silero-models

3.1.3 BLIP-2-based Visual Language Model Deployment

We establish a stable connection between our PC and the AI server. To be specific, we use the paramiko library in Python to establish an SSH connection. We use SFTP protocol to realize data transmission. The overall transmission is fast and stable. The detailed code is shown in Appendix A.

We deploy the Large Visual Language Model on the AI server. To be specific, we use the BLIP-2-based Visual Language Model for vision-language interaction. We would send real-time visual and audio data through the above mentioned connection to the server. Then the LVLM model would process it and send the updated answering messages back to my personal computer. The detailed architecture of the BLIP-2-based model[1] is shown in Figure 4.

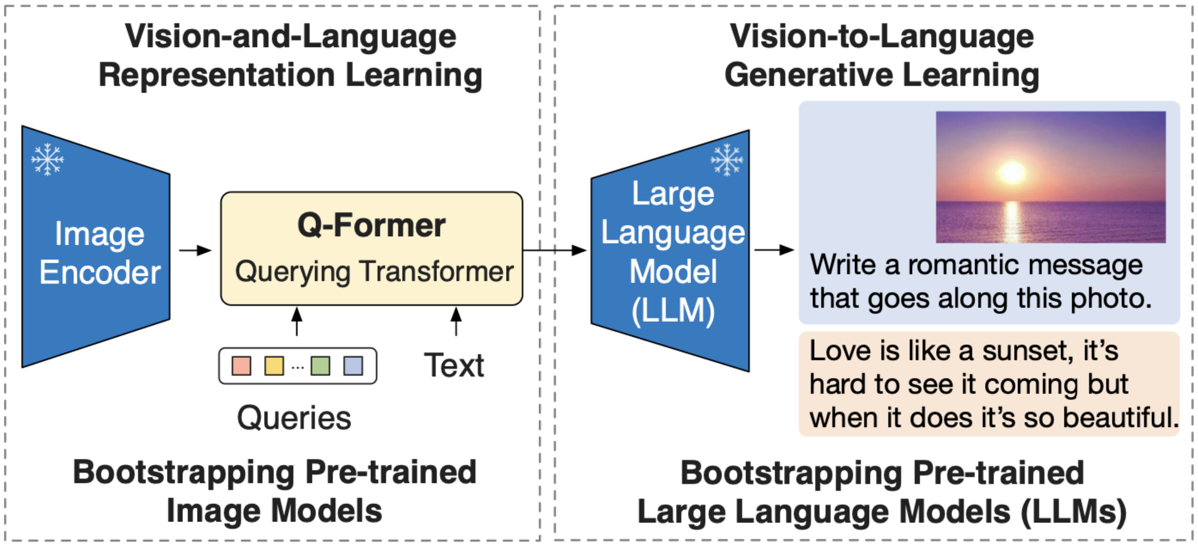

Vision-and-Language Representation Learning
Image Encoder
Q-Former
Querying Transformer
Queries
Text
Bootstrapping Pre-trained Image Models
Vision-to-Language Generative Learning
Large Language Model (LLM)
Write a romantic message that goes along this photo.
Love is like a sunset, it's hard to see it coming but when it does it's so beautiful.
Bootstrapping Pre-trained Large Language Models (LLMs)

Figure 4: Architecture of the BLIP-2-based Model

3.2 Raspberry Pi Auxiliary System

The integration of the Raspberry Pi Auxiliary System into our project was driven by the need to address significant challenges observed in the initial design, which relied heavily on a Vision Language AI model for controlling all operations. Here, we outline the reasons behind adopting the Raspberry Pi system to enhance functionality and efficiency.

Initial Design and Challenges: Our initial setup tasked the Vision Language AI model with guiding interactions entirely, from bottle placement to operation of the robot arm and water dispenser. This centralized approach led to considerable processing delays due to the model's inherent latency and added complexity, impacting system responsiveness and precision.

Rationale for the Raspberry Pi Auxiliary System: The adoption of the Raspberry Pi Auxiliary System was aimed at mitigating these issues through several key improvements:

1. **Reduced Latency:** By decentralizing control, the Raspberry Pi significantly cuts down on overall system latency, facilitating near real-time interactions between the user and the robotic arm.
2. **Simplified Processes:** The Raspberry Pi manages direct control over the robot arm and water dispenser, simplifying operations and allowing the Vision Language AI model to focus on providing high-level navigational aid rather than managing minute details.
3. **Enhanced Precision and Interaction:** With dedicated object detection and text-to-speech models, the Raspberry Pi system provides precise localization of the water bottle and clear auditory instructions, improving the guidance provided to users.
4. **Improved System Efficiency and Usability:** These modifications ensure smoother, more convenient operations and enhance the user experience, particularly in facilitating effective interaction for visually impaired users.

Introducing the Raspberry Pi Auxiliary System was a strategic choice to optimize our project's structure, enhancing operational efficiency, responsiveness, and user interaction. This approach not only improved system performance but also better aligned with our objectives of supporting visually impaired individuals in public spaces.

System Architecture Overview The Raspberry Pi Auxiliary System is strategically designed to enhance interaction convenience for visually impaired users. As shown in Figure 5, it integrates three cameras and a speaker to facilitate user interaction with the robot arm:

- **Two Web-Cameras:** Positioned to monitor the placement of the water bottle on the desk (both X and Y directions), ensuring it is within the operational range of the robot arm.
- **Pi-Camera:** Used to confirm the bottle's correct positioning at the water dispenser, ensuring accurate filling.
- **Speaker:** Provides real-time vocal instructions to guide the user in adjusting the bottle's placement.

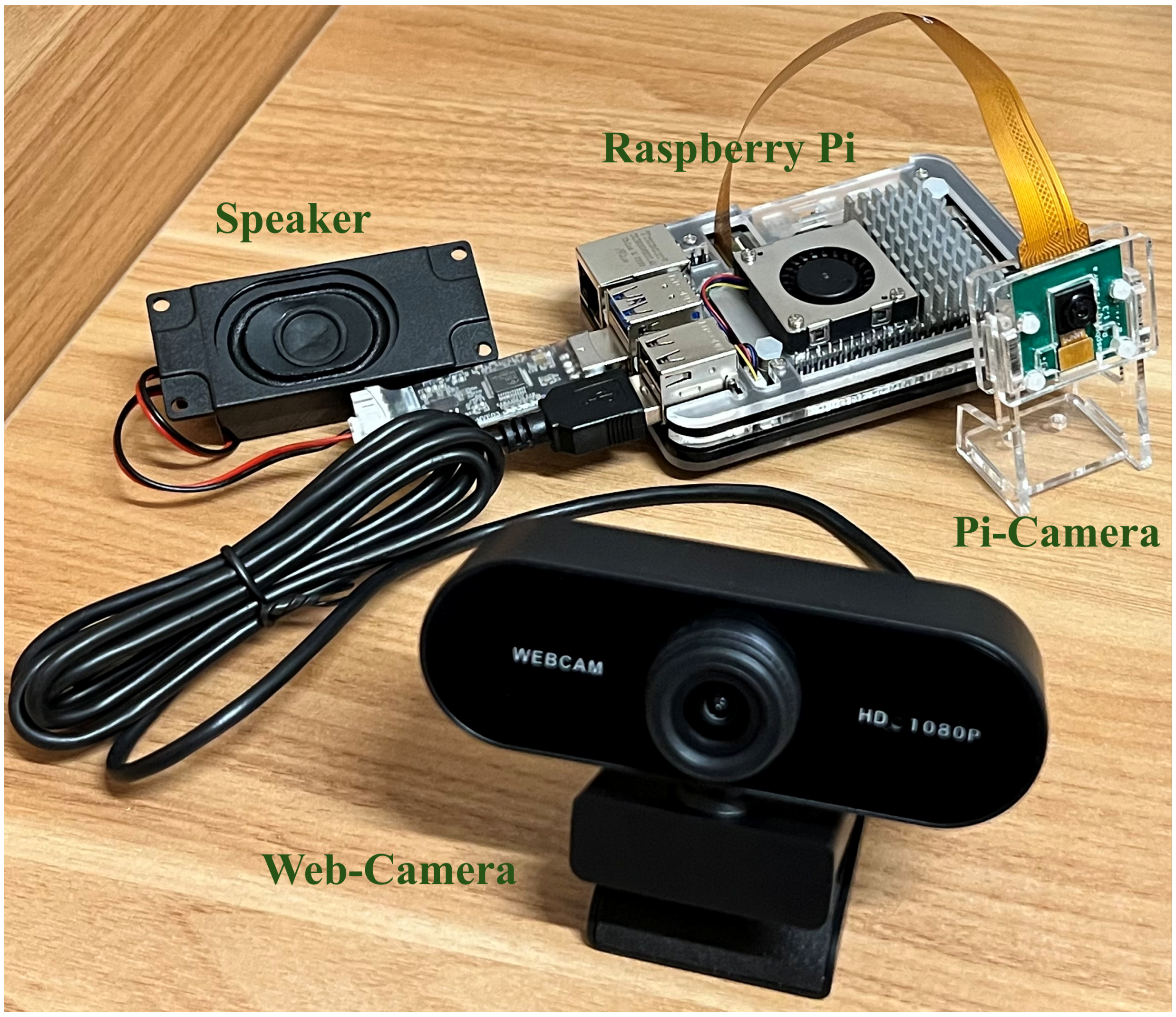


Figure 5: Architecture Overview of the Raspberry Pi Auxiliary System

3.2.1 Model Selection for Object Detection

For the task of detecting water bottles, we selected the **SSD-MobileNet-V2** model. This model is ideal for our edge device, the Raspberry Pi, due to its balance between speed and accuracy, making it well-suited for nearly real-time applications. SSD-MobileNet-V2 is designed specifically for mobile and edge devices, utilizing a streamlined architecture that maintains high accuracy while being computationally efficient. This efficiency is crucial in reducing latency and ensuring the system can operate in real time, which is essential for user interaction.

3.2.2 SSD-MobileNet-V2 Overview

The SSD-MobileNet-V2 [2] model is a highly efficient architecture designed for mobile and edge devices, combining the Single Shot MultiBox Detector (SSD) [3] framework with MobileNetV2 [4] . MobileNetV2 builds upon its predecessor, MobileNetV1 [5], by introducing an inverted residual structure with linear bottlenecks. This improvement over MobileNetV1 significantly enhances computational efficiency by maintaining depth information crucial for performance while reducing model size and computational cost.

Key Advancements of MobileNetV2 over MobileNetV1:

- **Inverted Residuals and Linear Bottlenecks:** Optimizes information flow within the network, essential for maintaining performance with reduced resource usage.
- **Streamlined Layer Transitions:** Reduces parameters and enhances computational efficiency, crucial for real-time applications on constrained devices.

By integrating the SSD framework, MobileNetV2 gains the ability to perform real-time object detection. SSD adds multiple convolutional feature layers, enabling effective multi-scale detection. This is vital for accurately spotting objects like water bottles in various positions and distances.

Benefits of MobileNet-V2 for the Raspberry Pi Auxiliary System:

- **Optimized for Edge Computing:** Tailored for devices like the Raspberry Pi, MobileNetV2 minimizes latency, essential for near real-time operations in our system.
- **Balance of Speed and Accuracy:** The combination with SSD ensures the system not only operates quickly but also maintains high detection accuracy, crucial for providing reliable guidance to visually impaired users.

This configuration of MobileNetV2 with SSD makes it exceptionally suitable for our Raspberry Pi Auxiliary System, providing an optimal balance between performance, efficiency, and usability in real-world applications.

3.2.3 Model Selection for Text-to-Speech

For converting text outputs into audible instructions, we chose the **eSpeak**[3] module. eSpeak is known for its simplicity, lightweight design, and wide language support, making it suitable for real-time speech synthesis on constrained devices. Its compact size and efficient speech generation capabilities ensure minimal impact on the system's resources, aligning with the need for a fast response time in our application.

[3] https://espeak.sourceforge.net/

3.2.4 Communication Setup with PCB Water Dispenser System

The communication between the Raspberry Pi and the PCB water dispenser system is established through GPIO pins. We utilize these pins to control LEDs on the PCB board, which indicate operational statuses like "waiting" and "dispensing." This setup allows for a straightforward, low-latency interface to manage the water dispensing actions based on the system's sensor inputs and camera detections.

3.2.5 Communication Setup with ROS System to Control Robot Arm

The procedures to set up and control the robot arm on the raspberry pi are as follows:

1. **Virtual Machine Configuration:** An Ubuntu VM (version 20.04) is set up on a PC using virtualization software.
2. **ROS Installation:** ROS Noetic is installed on the Ubuntu VM. This ROS version is compatible with Ubuntu 20.04 and provides the necessary packages and libraries to interface with robot hardware.
3. **Network Configuration:** To facilitate communication with the robot arm, the VM's network is configured to use a bridged adapter. This setup utilizes the Realtek PCIe GbE Family Controller, allowing the VM to connect directly to the same network as the host machine.
4. **Physical Connection:** Connect the UR3e Robot Arm to the PC using a network cable. This direct connection ensures reliable communication without latency issues.
5. **Automated Control Script:** On a Raspberry Pi, a shell script is developed to automate the process of controlling the robot arm. This script uses SSH to securely connect to the Ubuntu VM and execute ROS commands to operate the robot arm. The SSH setup ensures that commands can be sent remotely and securely to the VM where the ROS environment is configured.

3.2.6 Robot Gripper Control

The Makeblock Robot Gripper is operated via the MegaPi controller, which is programmed and interfaced with a Raspberry Pi. Initially, the necessary firmware is compiled and uploaded to the MegaPi. The Raspberry Pi is then connected to the MegaPi using a USB cable, and the Robot Gripper is attached to a DC motor port on the MegaPi. For controlling the gripper, the 'megapi' Python module is installed on the Raspberry Pi using pip. The motorRun() function within this module allows the Raspberry Pi to command the opening and closing of the gripper by controlling the motor's speed and direction, enabling precise manipulative operations.

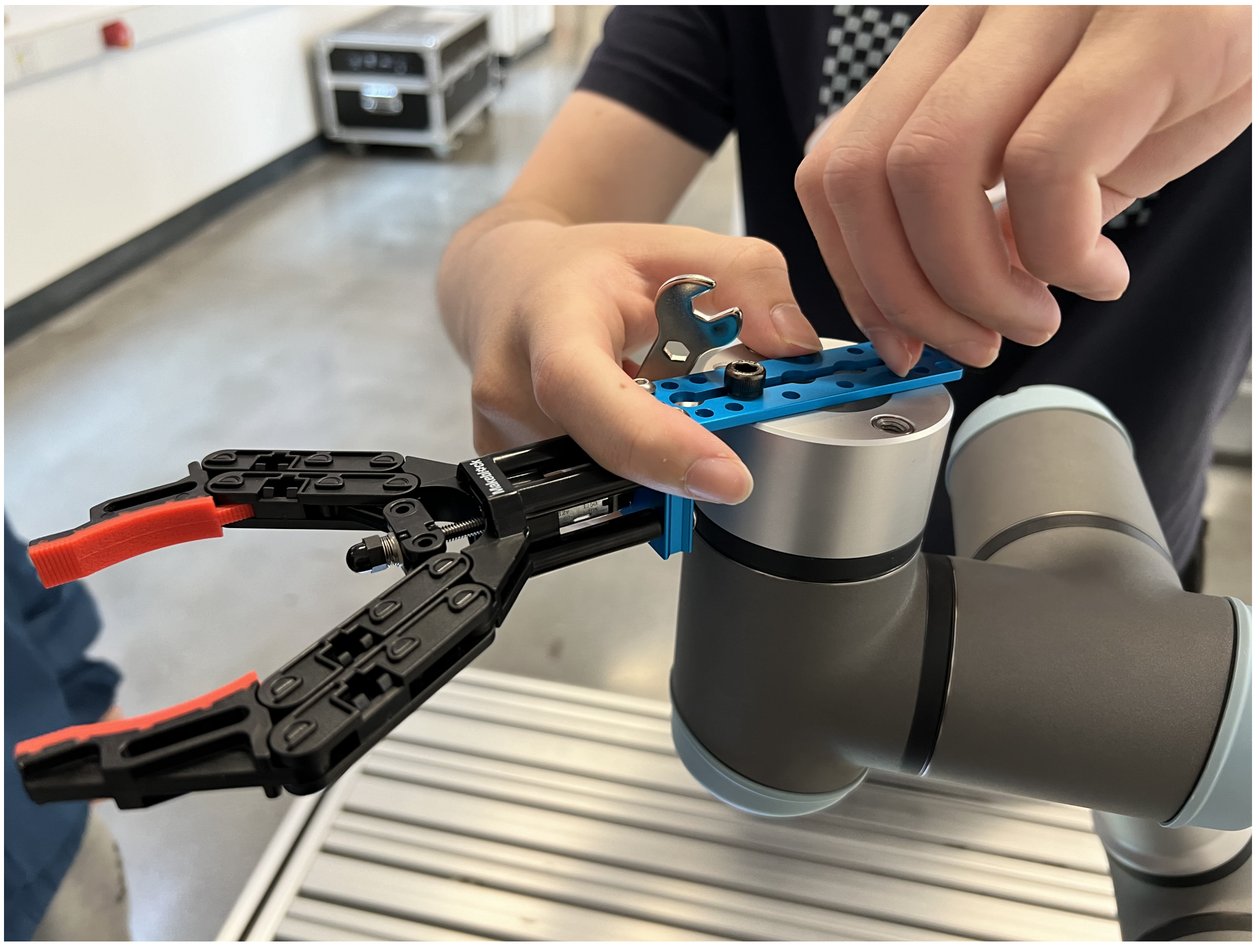


Figure 6: The Robot Gripper Mounted on the Robot Arm

3.3 PCB Water Dispenser System

The design details of our PCB board of the simulated water dispenser are shown in the following figures. On the board there are two LED lights with several gate-controlled logic. Based on the input from the Raspberry Pi Auxiliary System, when the water is being dispensed (green light on) and when the process is complete (red light on).

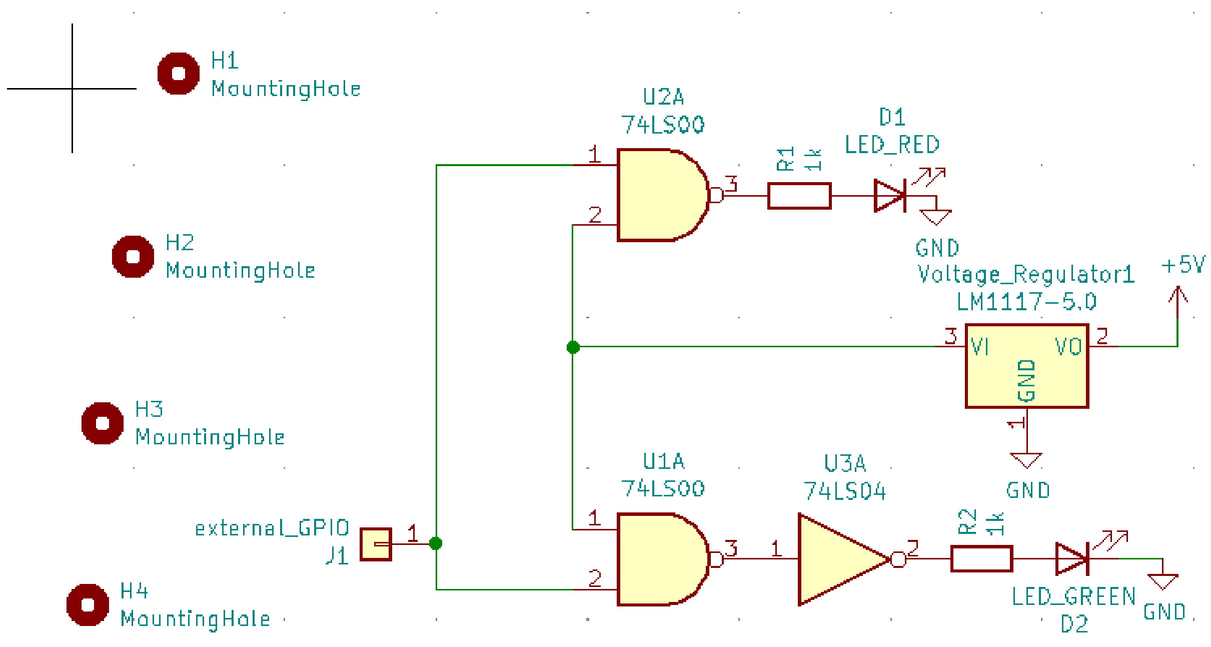
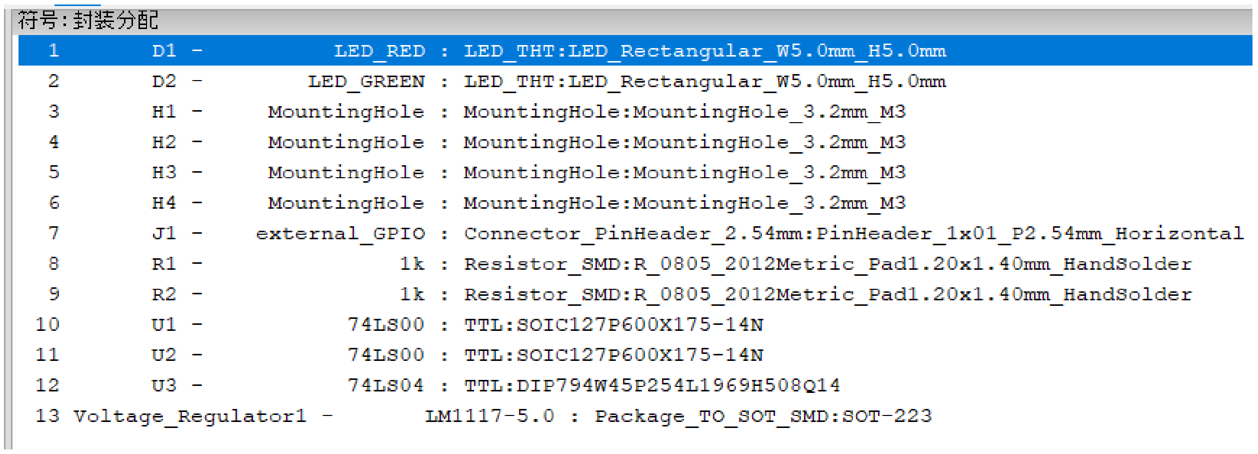


Figure 7: PCB Schematics



```
符号:封装分配
 1        D1 -        LED_RED : LED_THT:LED_Rectangular_W5.0mm_H5.0mm
 2        D2 -      LED_GREEN : LED_THT:LED_Rectangular_W5.0mm_H5.0mm
 3        H1 -   MountingHole : MountingHole:MountingHole_3.2mm_M3
 4        H2 -   MountingHole : MountingHole:MountingHole_3.2mm_M3
 5        H3 -   MountingHole : MountingHole:MountingHole_3.2mm_M3
 6        H4 -   MountingHole : MountingHole:MountingHole_3.2mm_M3
 7        J1 -  external_GPIO : Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Horizontal
 8        R1 -             1k : Resistor_SMD:R_0805_2012Metric_Pad1.20x1.40mm_HandSolder
 9        R2 -             1k : Resistor_SMD:R_0805_2012Metric_Pad1.20x1.40mm_HandSolder
10        U1 -         74LS00 : TTL:SOIC127P600X175-14N
11        U2 -         74LS00 : TTL:SOIC127P600X175-14N
12        U3 -         74LS04 : TTL:DIP794W45P254L1969H508Q14
13 Voltage_Regulator1 -     LM1117-5.0 : Package_TO_SOT_SMD:SOT-223
```

Figure 8: PCB Footprints Details

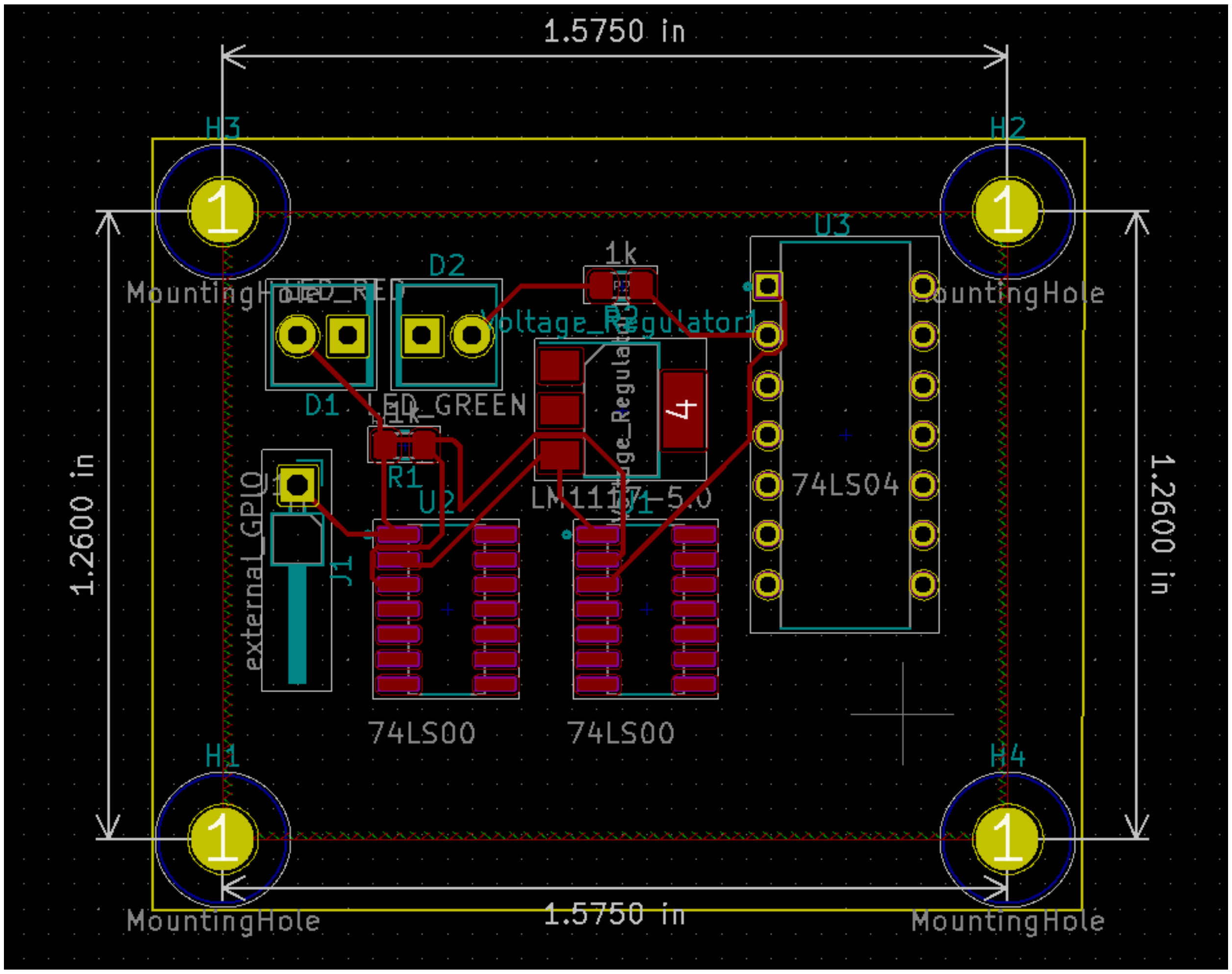


Figure 9: PCB Schematics in KiCad

3.4 Tolerance Analysis

A key design consideration is the latency in data transfer, which is critical to real-time interaction and control. We meticulously assess the latency focusiing on two main channels: user to computer, and computer to server.

User to Computer Data Transfer Analysis: A pivotal design concern is the latency during Bluetooth transmission of captured images and audio from head-mounted cameras and headsets to the computer. Assuming an operational distance of approximately 10 meters, we utilize the following formula to estimate Bluetooth transmission latency:

$$\text{Latency} = \frac{\text{Data Size}}{\text{Transmission Speed}} + \text{Propagation Delay}$$

Data Size is the total size of the data to be transmitted, measured in bits. Transmission Speed is the rate at which data is transmitted, measured in bits per second (bps). Propagation Delay is the time it takes for the signal to travel from the source to the destination, which can be calculated as the distance divided by the speed of the signal. However, for Bluetooth and similar short-range technologies operating at the speed of light, this delay is negligible compared to other factors.

Given Bluetooth 4.0's capability of up to 25 Mbps in high-speed mode and considering an average data packet size (1MB for a captured image), we can estimate the latency:

$$\text{Latency} = \frac{1 \times 10^6 \times 8 \text{ bits}}{25 \times 10^6 \text{ bits/sec}} = 0.32 \text{ seconds}$$

Computer to Server Data Transfer Analysis: Through simulations, we have estimated that data transfer delays between the computer and server can be confined to approximately 3-4 seconds. This latency is primarily influenced by network speed, server processing capabilities, and the data's complexity. Incorporating Python libraries like flash-attention has been instrumental in augmenting our AI models' processing speeds. These libraries enable more efficient handling of computations necessary for real-time analysis and decision-making based on the data received from user devices.

Conclusion: Experimental outcomes demonstrate that, despite variations, the entire processing duration stays within a few seconds, contingent on the complexity of the input data. This duration falls within our acceptable limits for real-time operations, underscoring the system's viability for responsive and effective user assistance. This analysis confirms our commitment to optimizing system performance while maintaining the real-time interaction that is vital for the success of our project.

3.4.1 Cost Analysis

Our fixed labor salary is estimated to be \$10/hour, and 50 hours for each person. The total labor costs for all partners:

$$4 \cdot \$10/\text{hour} \cdot 2.5 \cdot 50 \text{ hours} = \$5000$$

The costs of all parts in our project are shown in Table 1.

Part	**Cost**
Personal Computer (Macbook)	\$1200
Bluetooth Headset and iPhone Camera	\$1000
Raspberry Pi System	\$150
PCB Board (with Control Lights)	\$20
Robot Arm and AI Server (Borrow from ZJUI)	\$0

Table 1: Cost of Each Part

The grand total costs: $5000 + $1200 + $1000 + $150 + $20 = $7370.

4 Ethics and Safety

In the development of our robotics project, we rigorously adhere to the IEEE Code of Ethics [6] to uphold the highest standards of ethical practice and safety.

4.1 Ethics

Privacy (ACM 1.7: Respect the Privacy of Others) To protect privacy, we take strict measures to ensure the confidentiality and security of any personal information collected by the robot. We establish robust protocols based on industry standards to limit access to this sensitive data to authorized individuals with a legitimate need to access it. In addition, we carefully design and implement secure storage and processing procedures to reduce the risk of unauthorized disclosure or misuse. By prioritizing the protection of personal information, we demonstrate our unwavering commitment to maintaining the privacy and trust of individuals who interact with our robots.

Fairness (IEEE - Avoiding Real or Perceived Conflicts of Interest) The possibility of bias in the decision-making process of artificial intelligence is a major ethical issue. Recognizing this, we will strive to provide robots with a comprehensive understanding of human diversity and societal nuances through rigorous training and careful refinement. By exposing robots to a variety of data, including different demographics, cultural backgrounds, and environmental scenarios, we aim to equip robots with the ability to impartially discern and understand complex social dynamics.

Being Open (ACM 1.2: Avoid Harm) We are committed to ensuring full transparency in the robotics decision-making process. Our goal is to provide clear and understandable information to all stakeholders so that they can fully understand how the robot operates and the factors that influence its decisions. To achieve this, we keep detailed records of the algorithms, data inputs and learning methods used by the robot. Additionally, we are committed to an open approach to making information about the robot's functioning, including its training data, learning outcomes, and decision logic, readily available. By increasing transparency, we aim to build trust and confidence among users, stakeholders, and the broader community, thereby promoting ethical behavior by individuals or organizations when using our robotics.

Professional Development (ACM 2.6) Adhering to ACM's principles, our team dedicates itself to the continual enhancement of our knowledge and understanding of the societal ramifications of robotics. We recognize the dynamic nature of ethical standards and proactively refine our systems to stay abreast of new developments, ensuring that our robots serve as a benchmark for responsible AI and robotics practice.

4.2 Safety

Avoiding Accidents (IEEE - Priority to Public Welfare) Our robots are carefully designed with safety as a top priority to ensure that they do not jeopardize the personal safety of others or the safety of property. Equipped with advanced emergency stops and a range of sophisticated sensors, the robots are able to operate with increased vigilance, effectively preventing collisions with people and objects. These safety features are carefully designed to prevent accidental collisions and provide peace of mind in dynamic environments where human-robot interactions are frequent.

Staying Secure (ACM 3.7: Recognize the Need to Protect Personal Data) Given the advanced functionality and interconnectedness of our robots, it is critical to protect their integrity and guard against potential cyber threats. We are therefore building relevant security measures to strengthen its defenses and reduce the risks posed by malicious actors and cyberattacks. This requires the implementation of advanced encryption protocols, strict access controls and continuous monitoring mechanisms to detect and respond to any unauthorized attempts to compromise robotic systems or data. In addition, we prioritize regular security assessments and audits to identify vulnerabilities and weaknesses in our security infrastructure, enabling us to proactively address potential threats and ensure that our robots are resilient to evolving cyber threats.

Dealing with Mistakes (ACM 2.5 & IEEE - Acknowledge and Correct Mistakes) In the event of an unforeseen situation or error, the robot responds in a manner that prioritizes safety and reliability. The robot's operational framework incorporates fail-safe mechanisms and real-time monitoring capabilities to promptly identify and address any anomalies or deviations from expected behavior. By promptly notifying designated personnel or stakeholders of such occurrences, the robot facilitates rapid intervention to minimize potential risks and ensure continuity of safe and effective operations.

Responsibility (IEEE) In line with IEEE guidelines, our project is committed to the responsible deployment of robotics, ensuring they fulfill their intended roles effectively while safeguarding societal and environmental well-being. Our team maintains a vigilant approach to technology stewardship, regularly assessing and mitigating any negative impacts our robots may have, thereby ensuring our innovations contribute positively to society and operate sustainably within the environment.

Whistleblowing (ACM 1.4) Upholding ACM's ethical code, we foster an environment where whistleblowing is not just protected but encouraged, as it is crucial for maintaining the highest ethical standards. By promoting transparency and inviting scrutiny, we ensure any instance of misuse or ethical misconduct involving our robots is promptly addressed, reinforcing our commitment to integrity and the responsible use of technology.

References

[1] J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," *arXiv preprint arXiv:2301.12597*, 2023.

[2] Y.-C. Chiu, C.-Y. Tsai, M.-D. Ruan, G.-Y. Shen, and T.-T. Lee, "Mobilenet-ssdv2: An improved object detection model for embedded systems," in *2020 International conference on system science and engineering (ICSSE)*. IEEE, 2020, pp. 1–5.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[6] Institute of Electrical and Electronics Engineers. (2016) IEEE Code of Ethics. Accessed on: 2024-03-07. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html