ECE 445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

Remote Driving System

Team #17

BO PANG (bopang5@illinois.edu)
JIAHAO WEI (jiahaow4@illinois.edu)
KANGYU ZHU (kangyuz2@illinois.edu)

TA: Yi Wang

March 23, 2023

Contents

1 Introduction

1.1 Problem

Traditional chauffeuring service provides people with convenient and safe transportation solution. However, it also has many disadvantages. First, chauffeuring service may not be available in all areas, particularly in rural or remote locations. Also, chauffeuring service often requires advance booking, which can limit their flexibility, particularly for last-minute trips or changes in itinerary. Some companies may also have strict cancellation policies, which can be inconvenient for customers. In addition, the chauffeur might experience distraction within the car, which might raise safety issues.

With recent developments in communication technology and mixed reality, it is possible to build a remote driving system, in which a chauffeur can operate the owner's automobile remotely. This solution does not require the physical presence of chauffeurs. Thus, it offers more flexibility in terms of scheduling, as remote chauffeurs can operate the vehicle from anywhere at any time. This efficiency can reduce the cost of the service and make it more affordable to customers. Also, it is available in rural or remote areas as long as there exit stable network connection. Furthermore, it can provide an additional layer of safety as the chauffeur is not physically in the vehicle, reducing the risk of distraction or driver error.

1.2 Solution

Our remote driving system will provide real-time feedback of the car's external environment and information (e.g., car's current speed) to the remote chauffeurs. Through the use of advanced technologies, chauffeurs can remotely operate the automobile's movement using designated devices.

This system consists of four subsystems (MR Subsystem, Driving Control Subsystem, Server Subsystem, and Automobile Control Subsystem). MR Subsystem displays the information including car's external environment and status sent back by Automobile Control Subsystem. Driving Control Subsystem records commands of the chauffeur, which will be sent back to the automobile. Server Subsystem helps transmit message sent between these subsystems, achieving the result of remote driving.

1.3 Visual Aid

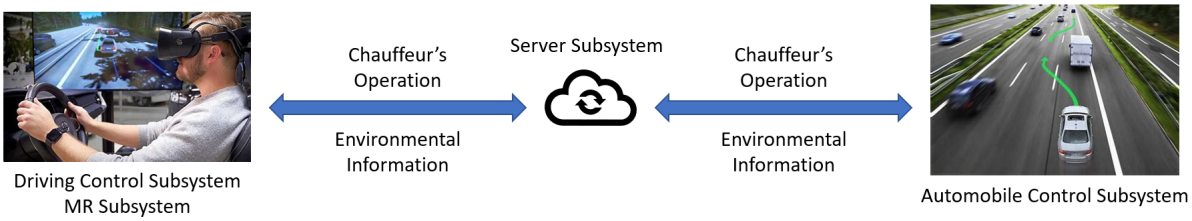

Figure 1: Visual Aid for Remote Driving System

1.4 High-level requirements list

We will substitute a TurtleBot3 (a programmable robot) for the automobile to reduce the complexity of the control subsystem.

- Under a chauffeur's operation, the TurtleBot3 should be able to navigate through the whole campus.
- The delay of the video sent back from the TurtleBot3 should be at least within 100 ms. The delay of the chauffeur's commands should be within 50 ms.
- The server can handle at least two connections from different chauffeurs. All chauffeurs can view the status of the TurtleBot3, and the one who will drive the TurtleBot3 will be selected according to a policy.

2 Design

2.1 Block Diagram

Our entire system consists of four subsystems (Hololens 2, Driving Control, Server, and TurtleBot3). The server receives, processes, and sends information to interact with the other three subsystems. In particular, Driving Control is mainly responsible for receiving and sending control commands. HoloLens 2 is responsible for presenting real-time information at the user end. TurtleBot3 Subsystem is capable of performing movement based on instructions and capturing real-time information of its environment.

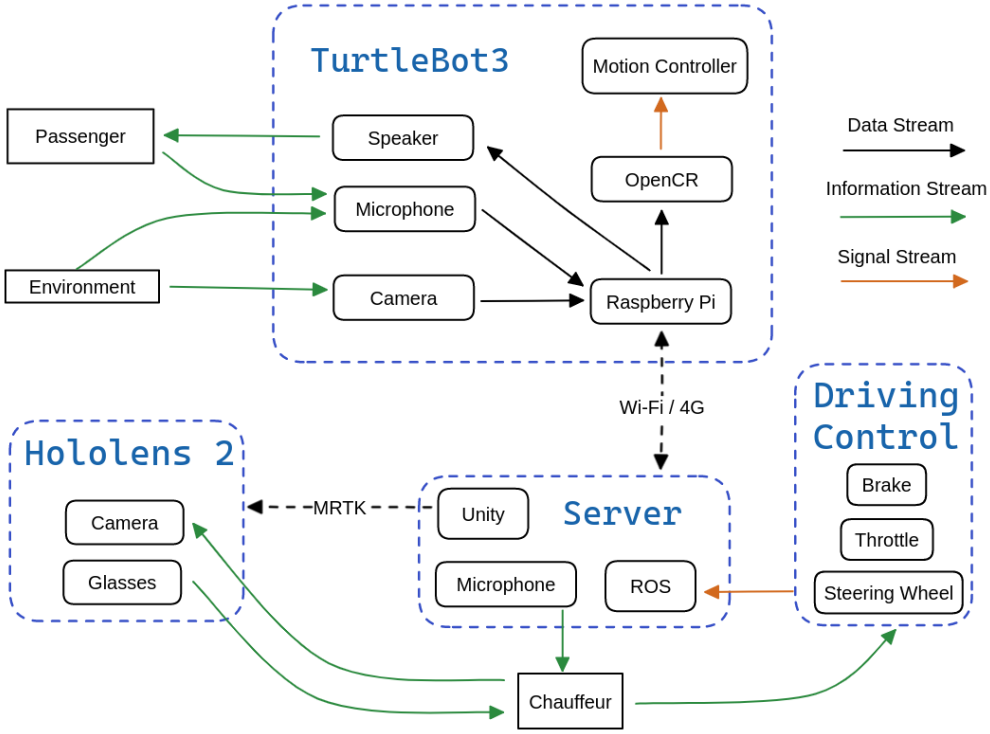

Figure 2: Block Diagram

2.2 Physical Design

Our physical design is shown in the following figure. The Turtlebot 3 consists of four layers, with the bottom layer containing the battery, the second layer containing the Raspberry Pi, the third layer containing the OpenCR, and the top layer containing the camera.

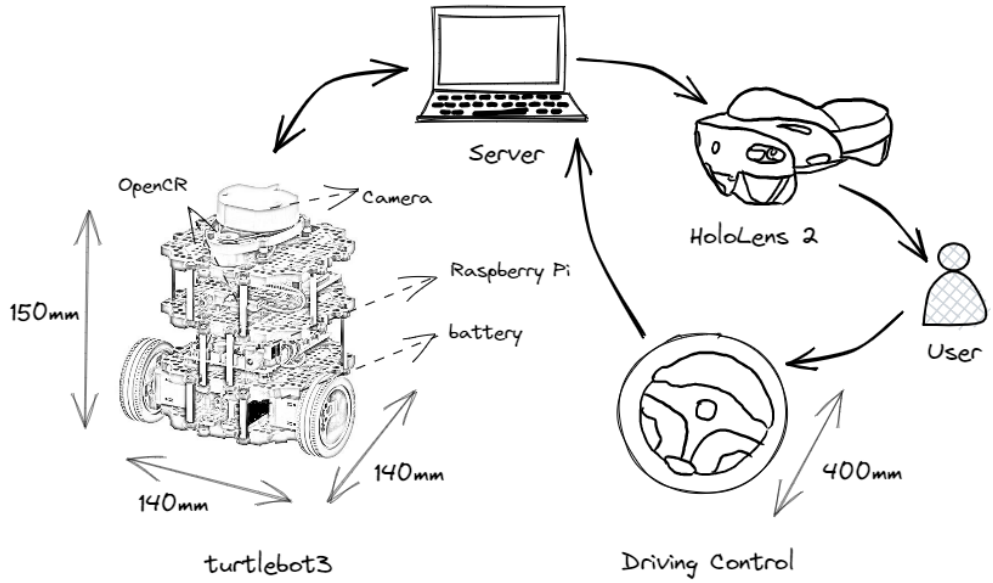

Figure 3: Physical Design

2.3 System Overview

In our remote driving system, a driver (the chauffeur) will wear HoloLens 2 glasses and remotely drive a car (TurtleBot3). The car's camera will transmit environmental information around the car to the server, which will synchronize the video information in front of the HoloLens 2 glasses. The driver will send control signals to the car's control board via Driving Control Subsystem. Automobile Control Subsystem will analyze the control signals and control the car's movement through the OpenCR control board.

The main platforms we use are Unity, Arduino IDE, and a OpenCR (Open-source Control Module for ROS (Robot Operating System)) development board. Unity is a comprehensive game development tool developed by Unity Technologies. It allows you to easily create interactive content such as 3D video games, architectural visualizations, and real-time 3D animations on multiple platforms. Through Unity, vehicle driving information can be displayed on HoloLens 2, thus increasing driving safety. Arduino IDE is used to control the OpenCR development board, which controls the hardware operation of OpenCR. The OpenCR is the main controller board for TurtleBot3 and is designed for embedded ROS development. The motion of the TurtleBot3 is controlled through the OpenCR development board.

2.4 TurtleBot3

The TurtleBot3 system needs to transmit video to the server. As driving requires low latency video, the User Datagram Protocol (UDP) is more suitable for our project than Transmission Control Protocol (TCP). We plan to use Socket and UDP for real-time video transmission and currently plan to use a resolution of 720p. We will use Socket and TCP

to establish reliable connections between the server and client. We will use OpenCV or Emgu CV (a cross platform .Net wrapper to the OpenCV image processing library) to capture camera images. Due to the 64kbit size limit of UDP packets, we plan to compress the images using JPEG format and send them using UDP protocol at the end.

Connection between TurtleBot3 Subsystem and other subsystems: A 360-degree camera on TurtleBot3 captures environmental information, which is then transmitted to the server via the OpenCR development board. The sound collector on TurtleBot3 captures surrounding audio information, which is transmitted to the server to facilitate communication between the driver and passengers. Meanwhile, TurtleBot3 also receives control signals from the server, which are used to control the vehicle's movement via the OpenCR and ROS operating systems.

TurtleBot3 is the most critical component in this project. It is used to control the vehicle through the OpenCR module on it. Development is done using the Arduino IDE, and remote communication is achieved by controlling information transmission and reception on the OpenCR development board. This part of the project requires us to understand the working principle of microprocessors and to develop their functions. OpenCR is the main controller board for TurtleBot3, designed for embedded systems development in ROS, providing complete open-source hardware and software. By understanding the ROS embedded system and processing the signals from the server, better control of TurtleBot3 can be achieved.

2.4.1 Sound and Raspberry Pi

The purpose of the sound system is to enable audio interaction between the car environment and the driver. Currently, we have decided to use the VOIP protocol for audio transmission. We will use a Raspberry Pi microphone, connected via a USB interface, for sound recording. It operates at a working voltage of 4.5V. For sound playback, we will use a Raspberry Pi speaker, powered via USB, with a 3.5mm audio jack.

Requirement	Verification
1. The microphone and speaker can be placed in a suitable location on the TurtleBots3, taking up a reasonable amount of space. 2. The sound system should enable stable remote connection between the car and the driver systems. 3. The system should have some degree of noise control capability. 4. The sound recording device should be able to collect sounds at 10dB or more, with an effective range of at least two meters, without interference from the speaker. 5. The speaker should be able to deliver clear sound that is easily heard and understood.	1.After completing the transmission function, sound recording should be tested to determine if the effective range of at least two meters is achieved. 2. The volume range of sound recording should be tested to determine if it can capture sounds at 10dB or higher. 3. The ability to transmit audio clearly in a noisy environment should be tested.

2.4.2 Camera and Raspberry Pi

The camera should be able to capture all environmental information required by the driver and transmit it to the server in a stable manner. The information collected by the camera is the most important information for driving. We plan to use two Raspberry Pi cameras with a viewing angle of 220 degrees each to collect video information, and transmit it to the server using the UDP protocol. The server will then stitch the video together to provide the driver with a panoramic view of the environment. To ensure stable video capture despite the movement of the car, we plan to use an image-stabilization algorithm.

Requirement	Verification
1. Stable connection between Raspberry Pi and Server, able to transmit video data reliably. 2. Implement a certain degree of shock absorption function to achieve anti-shake video during movement. 3. Capture the complete environment around the car, and the two cameras can synchronize transmission.	1.After the video transmission is completed, the delay of video data received by the Server should be within 100ms, and the delay between the two received video data should be within 40ms. 2. Test in a bumpy environment to ensure that the driver can obtain acceptable video information in a bumpy environment, to verify the correctness of the anti-shake algorithm implementation. 3. Able to splice video information on the Server-side, and the received video information can be spliced.

2.4.3 OpenCR and Movement

OpenCR (Open-source Control Module for ROS) is an open-source control module based on ROS (Robot Operating System). The module uses an ARM Cortex-M7 processor to communicate with ROS and achieve control of the robot. Currently, we plan to implement reliable communication between OpenCR and the server, receive and process control signals from the server, and convert the server's control signals into control signals for the car to achieve control of the car.

Requirement	Verification
1. Establish a reliable communication connection with the Server to transmit the control signals. The connection delay should not exceed 10ms. 2. Correctly receive and process various control signals sent from the Server and convert them into control signals of the car.	1. Test the communication delay between the OpenCR module and the Server to ensure that the delay does not exceed 10ms. 2. Send various types of control signals to the OpenCR module to verify whether it can accurately convert them into control signals of the car and control the car. Testing can be done using a simulation environment or an actual car.

2.5 Server

Server Subsystem plays the most important role in information exchange as the bridge for signal control. It needs to decode the video information sent by the TurtleBot3 and project it onto the HoloLens 2. It also needs to send signals from Driving Control Subsystem to the TurtleBot3 system to control the car.

2.5.1 Unity

Unity is a cross-platform game engine and development environment widely used in augmented reality (AR) application development. In this project, we need to connect Unity to HoloLens via the MRTK package, and at the same time, Unity needs to receive real-time information from the server (from the car, video information, data information, etc.).

Requirement	Verification
1. Connect HoloLens2 through MRTK. 2. Retrieve video information from the server. 3. Retrieve data information from the server sent by the distant car (e.g. car battery, speed).	1. Check the display on HoloLens2. If the displayed image on HoloLens2 is the same as the simulated image in Unity, and the simulated functions such as clicks can be performed in HoloLens2, then the verification is successful. 2. Check whether Unity can display the real-time image transmitted from the remote car camera, focusing on frame rate and latency. If there is no disconnection or frame dropping for a long time, the verification is successful. 3. Continuously change the running status of the car, and check whether the data information about the car in Unity changes dynamically accordingly.

2.6 HoloLens 2

The HoloLens 2 system carries all the learning that the driver can obtain. The Server will project the transmitted video signal onto the HoloLens 2, allowing the driver to see the environment around the car from multiple angles. Through the Unity platform, the running information of the car can be projected onto the HoloLens 2 for the driver's reference. With Hololens 2, We need to ensure that the driver can see a clear video captured by the remote turtlebot 3's camera virtually located at a distance of 1 m from the driver. The field of view in front of the driver should be approximately 120 degrees from left to right. They should be able to see more angles when turning their head.

2.6.1 Camera

Hololens 2 has four environment-aware cameras and two depth cameras. It can achieve functions such as spatial awareness and gesture recognition. Through programming, we need to enable these cameras to capture gestures and enable interactions between hands and objects (such as clicking, popping up dialog boxes, etc.).

Requirement	Verification
1. Recognize gestures and display interactive menu bars when clicking on objects (e.g. small balls). 2. Continue clicking on different buttons on the menu bar to trigger multiple functions (e.g. display navigation maps, play music, etc.).	1. Adopting the method of multiple verifications, first run the interface on the Unity platform to virtually verify whether the small ball can be successfully clicked. After wearing the HoloLens 2, click on the small ball from different angles multiple times, conduct more than ten trials for each angle, and ensure a success rate of over 95 2. First, run the interface on the Unity platform for virtual verification. After wearing the HoloLens 2, click on each button multiple times (no less than 10 times for each button) to ensure that each button triggers the correct function with a success rate of over 95

2.6.2 Glasses

Hololens 2 glasses serve as a display device for mixed reality experiences. They enable users to see and interact with virtual information that is overlaid onto the real world, creating a mixed reality environment. With such glasses, we need to provide the driver with a clear and stable field of view. Specifically, the screen should be positioned at a distance of 1 m from the driver, and the immersive virtual interface should be located directly in front of the driver.

Requirement	Verification
1. It can display a stable immersive virtual interface (e.g. dashboard, driving environment) in front of and below the user (at a close distance), and the alignment of the immersive virtual interface can be reset when the driver changes position. 2. The screen displaying the remote view needs to be at an appropriate distance from the driver so that they can see the content clearly. The distance is set to about one meter, and the alignment of the screen displaying the remote view can be reset when the driver changes position. 3. It can provide drivers with a stable view of the remote driving environment (captured by the car camera), with the interface displayed in a curved shape, surrounding the driver's field of view (approximately 124°). When the driver turns their head, they can see the surrounding environment of the car (environmental information of about 180° on the left and right sides).	1.When the driver is driving or slightly moving, check whether the immersive virtual interface is fixed. Move the driver's position several times and check whether it can be successfully reset each time, and whether the position of the interface reaches the front and bottom of the user. 2. When the driver is driving or slightly moving, check whether the screen displaying the remote view is fixed. Verify the distance between the screen and the driver by first recording the current position of the driver, then having the driver pass through the screen, and finally measuring the distance between the two recorded positions to see if it is 1 m. 3. Ask the driver to look straight ahead, and have someone stand about 120 degrees off the driver's front, ensuring that the driver can see the scene from this angle in the screen. When the driver turns their head, make sure they can see the left and right visual areas that cannot be seen when looking straight ahead (about 180 degrees away from the front), which means the verification is successful.

2.7 Driving Control

Driving Control Subsystem is the control system for the chauffeur, which should include components such as a steering wheel, accelerator, and brake. The driver sends control information to the server by using the corresponding control parts, achieving remote control of the car.

Connection between Driving Control Subsystem and other subsystems: The driver

controls the car by watching the audio-visual feedback projected by HoloLens 2 Subsystem. The driver uses the steering wheel, accelerator, and brake pedals in Driving Control Subsystem to control the car. Driving Control Subsystem is directly connected to the server, which receives the control signals from Driving Control Subsystem and transmits them remotely to the car, enabling remote control of the car.

The role of Driving Control Subsystem in the project is to provide a more immersive and realistic remote control experience. By using a driving interface similar to real life, the remote driving process becomes more accessible and safer.

2.7.1 Steering controller

The steering wheel system is used to provide feedback on the steering signals made by the driver. We are planning to develop a force feedback steering wheel control system to enhance the driver's driving experience, and to better simulate the driving environment and reduce the discomfort of driving.

For the steering wheel control system, we need to simulate the real driving experience. The force feedback needs to accurately simulate the real driving sensation, including the steering force of the steering wheel and the feedback from the vehicle suspension system to the steering wheel. At the same time, good force feedback should be smooth and continuous without noticeable jitter or sudden changes. We will also ensure low feedback latency to enable safer control.

We need to develop the driver ourselves to make the steering wheel system's signals recognizable by the server. We are currently attempting to use Pygame to implement the driver. Specific implementation details are yet to be determined.

Requirement	Verification
1. Develop a suitably sized force feedback steering wheel simulator that accurately simulates the real driving experience and provides accurate steering wheel turning forces. 2. The latency between the steering wheel simulator and the server must be within 10ms and provide low latency control. 3. The driver must accurately recognize the steering wheel movement angle.	1.Calculate the feedback torque of the feedback motor to ensure that the torque is within a safe and reasonable range. 2. Test the latency of the server driver detection signal, which must be less than 5ms. 3. Check whether the feedback signal is reasonable and can truly reflect the driver's actions.

2.7.2 Accelerator and Brake Pedal

The accelerator and brake pedal system is responsible for transmitting the throttle and brake information of the vehicle, which is processed by the driver's pressing degree. It can provide feedback to the server about the driver's operation. We plan to use Pygame to implement the driver to obtain control information from the pedal system.

Requirement	Verification
1. Able to accurately feedback the driver's driving operation. 2. Can simulate the throttle and brake in real life to some extent. 3. The driver can accurately feedback the pedal system signal to the server.	1.Calculate whether the relationship between the pedal signals obtained by the driver and the human stepping force is reasonable. 2. Test the signal delay detected by the server driver, which must be less than 5ms.

2.8 Schematic

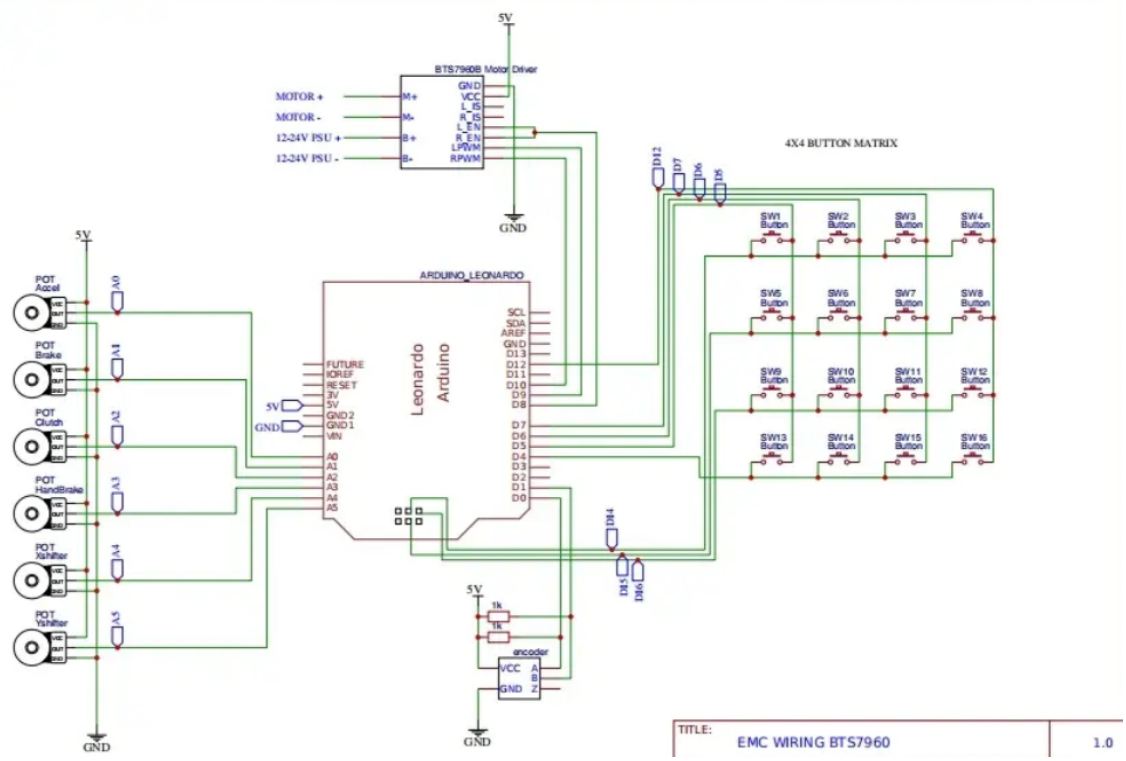

Figure 4: Driving Control Schematic

2.9 Tolerance Analysis

2.9.1 Bandwidth Requirements

A critical feature of this project is reliable and timely transmission of video data captured by the camera in TurtleBot3. The following analysis shows that the network condition is sufficient for our video transmission.

Since our project will be used inside campus, the router with IPv4 address 10.105.254.214 is used to measure the network bandwidth. This router is chosen because it appears to be the gateway router of school network. We used `tracert` to examine the routes of data packets. Although the bandwidth usage varies, we measure the round trip times at different time during a day and take the maximum as the upper bound of network delay. We used `ping` to measure the round trip time, and the result is 9ms, as indicated in the figure. Based on this result, we estimated the upper bound of transmission delay from our server to TurtleBot3 as 20ms.

```
Pinging 10.105.254.214 with 32 bytes of data:
Reply from 10.105.254.214: bytes=32 time=7ms TTL=253
Reply from 10.105.254.214: bytes=32 time=9ms TTL=253
Reply from 10.105.254.214: bytes=32 time=4ms TTL=253
Reply from 10.105.254.214: bytes=32 time=6ms TTL=253

Ping statistics for 10.105.254.214:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 9ms, Average = 6ms
```

Figure 5: Output of ping 10.105.254.214

The eduroam Wi-Fi utilize 5.0 GHz channels. Assume a Wi-Fi access point covers at most 100 people, and a 5.0 GHz Wi-Fi connection can provide 1300 Mbps upload and download speed. In the worst case, the upload and download speed for our TurtleBot3 is 13 Mbps.

The camera we used to capture the environment of TurtleBot3 has a sensor with 1080 pixels. We use JPEG image compression for video data and UDP for data packet transmission. Assume each pixel need 3 btyes to represent. With a rate of 360p video frame every 1/30 second, the transmission time needed for each frame is

$$480 * 360 * 3\,B/13\,Mbps + 20\,ms \approx 58.03\,ms$$

which is within our high-level requirement.

To make sure Wi-Fi has sufficient signal strength, we used RSSI (Received Signal Strength Indicator) in dBm to indicate the signal strength. The following of table shows RSSI for

possible places where our TurtleBot3 will be driven through. These places have strong Wi-Fi signal for maintaining a good network connection.

Place	RSSI (dBm)
Plaza	-48
Lecture Theatre East	-41
Lecture Theatre West	-46
Auditorium	-51
Resident College 2	-72
Resident College 3	-71
Library	-52
North Teaching Building	-59
North Circular Road	-70

Table 1: RSSI of Landmarks in Campus

3 Cost and Schedule

3.1 Cost Analysis

- Labor
  Assume salary per hour is ¥45. Then the total labor cost is $45 \times 3 \times 85 = 11475$ ¥.
- Bill of Materials

Name	Description	Price	Qty	Total
Pi Camera	wide-angle camera	¥40	2-3	¥100
Speaker	Normal Speaker	¥15	1	¥15
Microphone	Normal microphone	¥9	1	¥9
Power supply	24V20A500W	¥88	1	¥88
Belt	HTD3M480*15MM	¥13	1	¥13
Motor	MY1016 24V300W	¥65	1	¥65
Development Board	Leonardo_R3	¥26	1	¥26
Steering wheel		¥68	1	¥68
DC motor drive board	4f0W DC motor drive board	¥52	1	¥52
Pedals	Normal Pedals	¥18	2-3	¥45
Base support	Just normal wood and iron	N/A	1	N/A
Screws and nuts	Just normal screws and nuts	N/A	1	N/A
Total				¥481

- Total
  The grand total of costs is 11956 ¥.

3.2 Schedule

Week	Bo Pang	Jiahao Wei	Kangyu Zhu
3/27	Learn OpenCV programming under the C++ framework. Choose appropriate encoding and decoding methods.	Program OpenCR to manipulate the motion of TurtleBot3.	Establish a connection between Unity and HoloLens.

Week	Bo Pang	Jiahao Wei	Kangyu Zhu
4/3	Roughly completed the framework for UDP video transmission. Build a PCB architecture based on the working principle and complete component procurement.	Understand the working principle of the steering control system, and main control solutions.	Complete the video streaming from the server into Unity for rendering.
4/10	Complete the framework for UDP video transmission function.	Brainstorm the working plan of the driving system.	Implement video stitching for multiple camera images.
4/17	Understand the working principle of firmware and flash the firmware. Print PCB and solder components.	Understand the working principle of the driving system, and attempt to program to drive the analysis of the steering wheel motion signal.	Optimize the display of remote videos in Unity, including features such as following movement of head rotation.
4/24	Install cameras in TurtleBot3 and make Raspberry Pi collect video signal.	Understanding how firmware works and burning firmware.	1. Transmit information such as battery level and speed of the turlebot to Unity. 2. Model AR components in Unity and display real-time information such as battery level and speed.
5/1	Setup Raspberry Pi and establish connection between Raspberry Pi and the server.	Complete the driving programming and transmit the control signal in real time.	Collect external audio using a microphone on the car and transmit it to the server.
5/8	Conduct testing.	Conduct testing.	Transmit the microphone signal from the server to the car radio.

4 Ethics and Safety

Due to safety considerations, our project involves a variety of potential hazards that may arise during the interaction between humans and machines.

- The potential risks associated with using lithium batteries on Turtlebot 3.

  Turtlebot 3 is powered by lithium batteries. Although it is environmentally friendly, it has poor safety and there is a risk of explosion. To protect the lithium battery and reduce safety risks, we have established the following agreements regarding the use of Turtlebot 3 and its lithium battery:

  1. Avoid overcharging and overdischarging: Overcharging or overdischarging can damage its performance and shorten its lifespan. Charging and discharging should follow the Turtlebot instructions charging guide.
  2. Avoid excessive vibration or violent collisions: Lithium batteries should not be subjected to excessive vibration or violent collisions because this may cause internal short circuits, resulting in battery overheating, fire or even explosion. Therefore, the activity area of the robot should be on flat ground.
  3. Place the robot in a dry environment because placing it in a humid environment may cause a short circuit in the battery, causing damage to the battery.
  4. When removing and replacing the lithium battery, handle it gently and turn off the switch when not using the robot.

- The precision of the control system

  The instructions given by remote driver are transmitted to the car, allowing them to control its movements remotely. However, inaccuracies or delays in the control system can pose serious safety hazards for both the vehicle and its carring passengers. For instance, on narrow or busy roads, even minor inaccuracies in the control instructions could result in severe accidents or collisions. To minimize these risks, it is imperative that the human-machine interaction system has high control precision. The remote driver's controller should be designed to have similar function and precision to that of a physical steering wheel and provide a precise and responsive control experience. One of the efficient ways is to add a force feedback functionality (simulated vibration, resistance) to the steering wheel so that users can perceive physical changes. This will enable the remote driver to make more accurate and effective driving decisions, especially in critical situations.

- The mental and physical state of the remote driver

  Remote driving system requires designated drivers to wear virtual reality devices, such as glasses, which often have relatively large sizes and weights compared with the normal devices in our everyday life. These devices could possibly cause distraction to the remote driver and increase their susceptibility to fatigue. Moreover, extended use of virtual reality devices can lead to symptoms such as dizziness, significantly compromising the safety of remote driving. These are all factors that could possibly violate "to hold paramount the safety, health, and welfare of the public, to strive to comply with

ethical design and sustainable development practices" in IEEE code of ethics[1].. To avoid this from happening, we we highly recommend that designated drivers undergo rigorous screening and training before operating this system. Additionally, they must take sufficient rest after driving for an extended period. These steps are essential to guarantee the safety of both the driver and passengers. We prioritize this issue and will continuously work towards improving the system to reduce potential risks.

- Communication delay, stability and resolution of the transmitted video

  The system use cameras to capture the driving situation and transmit it to a remote driver's location. Through virtual reality technology, the substitute driver can interact with the vehicle in real-time, enabling them to navigate the roads and ensure the safety of passengers. However, this technology is not without its challenges. Accroding to ACM code of ethics: "the emergent properties of systems should be carefully analyzed[2]." In our system, the transmission of the live video feed from the car to the remote location may be unstable and delayed, leading to potential safety risks. Furthermore, areas with weak signals may require a reduction in video resolution, which could further compromise the remote driver's ability to make informed judgments and decisions. To address these issues, it is essential to focus on enhancing the hardware of the cameras to improve their maximum resolution. In addition, optimizing the transmission network can help to enhance stability, reduce latency, and minimize delays during transmission. In our project, we propose to use 5.0 GHz channels in our campus, with JPEG image compression for video data and UDP for data packet transmission to ensure reliable and timely transmission. By addressing these challenges, we can improve the effectiveness and safety of driver substitution, ensuring a seamless and secure driving experience for all passengers.

References

[1] IEEE. “IEEE Code of Ethics.” (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html.

[2] ACM. “ACM Code of Ethics and Professional Conduct.” (2018), [Online]. Available: https://www.acm.org/code-of-ethics.