ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Augmenting Virtual Reality (VR) with Smell

Team #15

BAOYI HE (baoyihe2@illinois.edu)
KAIYUAN TAN (kt19@illinois.edu)
XIAO WANG (xiaow4@illinois.edu)
YINGYING LIU (yl73@illinois.edu)

Sponsor: Rakesh Kumar
TA: Qi Wang

May 10, 2023

Abstract

The absence of scent in virtual reality (VR) experiences limits the immersive potential of these technologies, preventing users from experiencing a full sensory experience. Our virtual reality (VR) system aims to provide a more immersive and realistic experience for users by integrating scent cues into the virtual environment. In this report, we demonstrate that with virtual reality (VR) hardware, an Arduino micro-controller, a well-designed circuit that controls the scent emitter, virtual reality (VR) software to generate virtual scenes, and a scent simulator to generate correct scents information corresponds to the given virtual scenes, we successfully implemented all high-level functionalities of the system.

Contents

1 Introduction

1.1 Purpose

Virtual reality (VR) technologies are rapidly growing and becoming more prevalent in our daily lives. However, these technologies have not yet fully addressed the sense of smell, which is a critical aspect of human experience. The absence of scent in virtual reality (VR) experiences limits the immersive potential of these technologies, preventing users from experiencing a full sensory experience. There are several challenges associated with integrating smell into virtual reality (VR) systems. One of the main challenges is replicating scents accurately and consistently. Unlike visual and auditory cues, scents are complex and difficult to reproduce. Additionally, the delivery of scent cues must be carefully controlled to ensure that users are not overwhelmed or nauseated.

To solve the problem, our team proposes a solution, which incorporates hardware and software components that can simulate various scents in real-time, in response to events in the virtual reality (VR) environment. As shown in Figure 1, our device is composed of three parts physically: one virtual reality (VR) headset, one neck scent-emitter which is responsible for emitting scents, and one perception module for collecting environmental information such as wind speed and room size. The headset needs to be connected via DisplayPort 1.2 and USB 3.0 to a computer to load the virtual scenes we built. The sensors on will be connected to the computer via Arduino. When the user enters our virtual world, he will see our carefully designed game scenarios including a 3D writing program, drinking tea, plucking apples from a tree, a farmland, and a garden. When the user gets closer to the objects that have a special odor, the scripts attached to those objects in the virtual world will start to transmit distance information to our software. Based on physical models, our scent simulator will simulate the scent intensity and duration the scent emitter should have, and then transmit this information to it. Finally, the user will smell the corresponding scent emitted by the scent emitter. Besides, we also design an interaction module, where a guest of wind in the real world can generate virtual wind in the virtual world, that is, the flowers and mills will sway in the wind with the sensed wind speed.

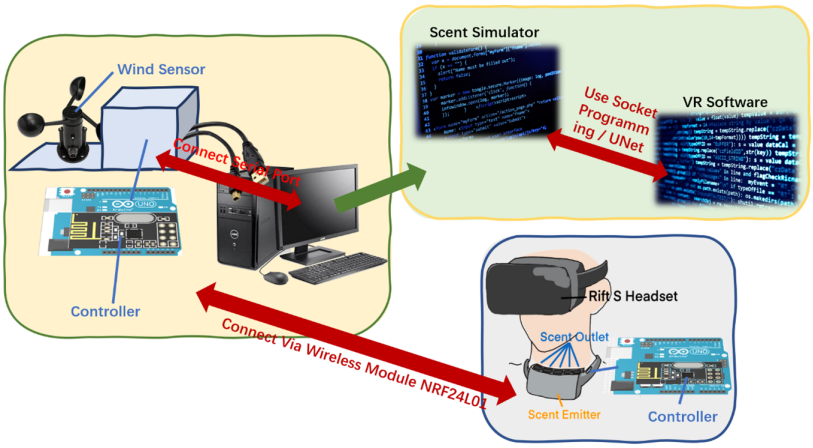

Figure 1: Overview of our solution

1.2 Functionality

Virtual Reality (VR) Software and Hardware Deliver a smooth and immersive demo at a minimum of 90 FPS. This allows users to have a complete and immersive experience in the virtual world. Incorporate diverse scenarios to showcase the integration of olfactory elements in VR scenes. There should be 0 latency when switching into different scenarios. This provides users with an interesting gaming experience and adds more fun to our project.

Scent Simulator Software Employ suitable physical models for scent simulation, incorporating valid modifications and assumptions. This allows users to have sensory experiences like in the real world. Accurately map the relationship between distance and scent intensity for a seamless and natural user experience. One object should be matched to exactly 1 scent without mistakes. This also allows users to have sensory experiences like in the real world.

Scent Emitter The frequency of the oscillator is between 108 and 110 Hz. The overall frequency should be between 0.12 and 0.13 Hz for low-level, between 0.24 and 0.26 Hz for medium-level, and no-gap oscillating for high-level. This ensures the stable emission of various scents. Select distinct and appealing scents for differentiation. Combinations of different scents should be available. The weight of the neck scent-emitter will not exceed 0.3 kilograms, making it easy for users to carry around during virtual reality (VR) experiences, and have less stress on users' necks. This design enables the scent emission device to be lightweight, comfortable, and aesthetically pleasing.

External Perception Module The wind speed should affect the virtual scenarios. The wind sensor must be able to detect wind speed in the range of 1m/s to 60m/s for the resolution ratio to be lower than 0.2m/s to fit our requirements. The wind sensor should transmit wind speed to the computer smoothly with a speed of at least 100Kbps. This allows our real-time detection information of wind speed to be transmitted to the software with little latency and thus allows users to have an immersive experience.

Communication between Components Maintain sufficient bandwidth of no less than 2Mbps for real-time inter-component communication. This allows communication among three parts to be fluent, and latency to be minimized. Ensure communication doesn't obstruct user movement or interaction with VR scenes. The overall reaction time should be no more than 300 milliseconds. This includes the scent simulator's calculation time and data transmission time between different protocols.

1.3 Subsystem Overview

Our whole device consists of five subsystems, including two software subsystems: a virtual reality (VR) software to generate virtual scenes and a scent simulator to generate correct scents information corresponding to the given virtual scenes. To be more specific, the scent information includes scent species, scent intensity, and scent duration. The **virtual reality (VR) software** enables us to build engaging scenarios, tell users interactive stories, and transport people to new worlds by building virtual reality experiences, which allows our device to fulfill the most important high-level requirement–immersive experience. This subsystem also needs to pre-process users' movement information and head orientation collected by virtual reality (VR) hardware and transmit the information to scent simulation software for further calculation. The **scent simulation software** gets the pre-processed data from virtual reality (VR) software and uses our efficient and effective algorithm to decide which scent to emit, how long the scent last, and how strong the smell is. This information will then be transmitted to the scent emitter subsystem for the release of real scent.

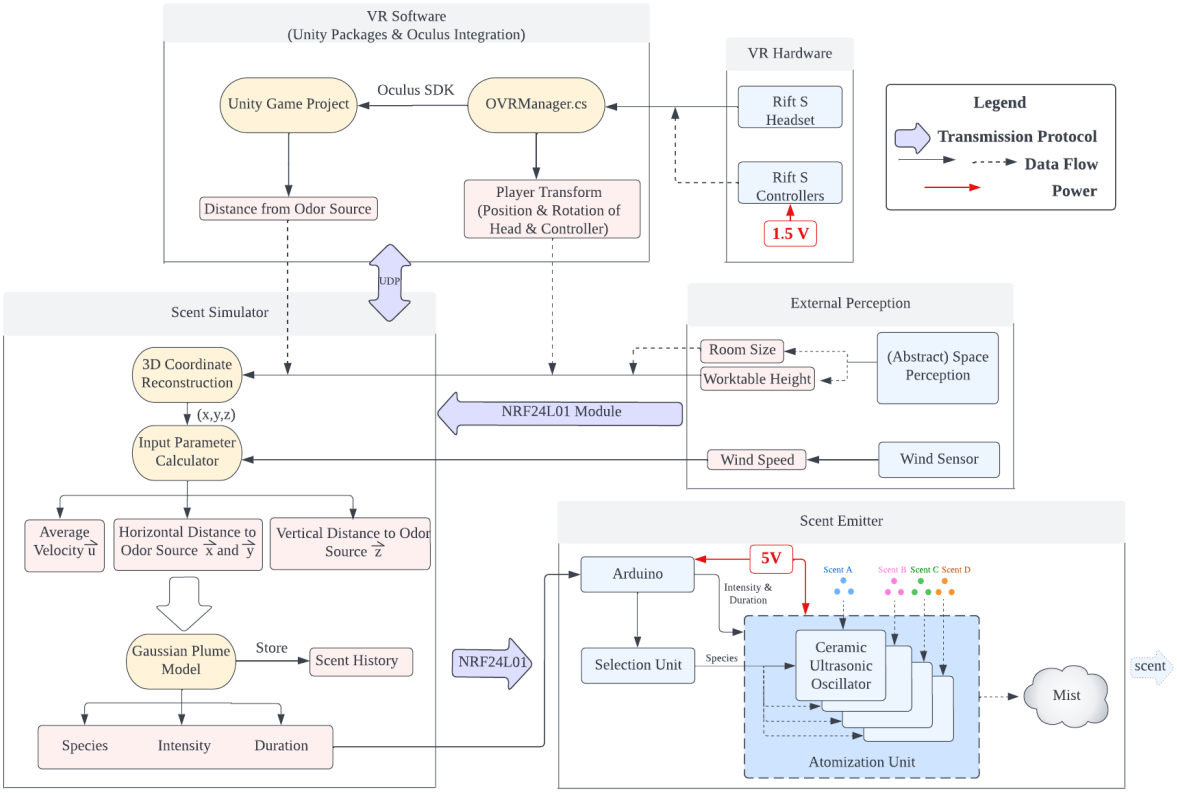

Figure 2: Block Diagram

The remaining three subsystems incorporate hardware components: the virtual reality (VR) hardware for users' motion capture and model rendering, a scent emitter to emit scents based on information got from the scent simulator, and an external perception module to collect environment information such as room size and wind speed. The **virtual reality (VR) hardware** provides the hardware basis for virtual reality (VR) software and uses Rift S as the key hardware component. It consists of a virtual reality (VR) headset and two controllers. Through the headset, the user (as the camera inside the project) can see virtual scenes. The **scent emitter** consists of a scent cartridge, an Arduino module, a selection unit, and an atomization unit with several ceramic ultrasonic oscillators. **The external perception module** is composed of two parts: one wind sensor and one space perception module. Wind sensors collect wind speed information and then transmit them to virtual reality (VR) software using a wireless connection protocol. The reason we desire to collect the room size information is that the diffusion of scents is highly related to the room's condition.

Two software run on the computer, where **data transmission** is handled by process communication protocols. The scent emitter and external perception module are integrated together into a neck scent emitter, which users can hang on their necks. The VR headset and the computer are connected via DisplayPort1.2, where the virtual scenes will be

loaded via the wired connection. The scent-emitter and external perception module use a wireless communication module—NRF24L01 to talk with the software on the computer. The novelty of our proposed solution is that we not only emit scents based on corresponding scenes but also integrate environment information in our virtual scenes. For example, when users use our device outside, they may feel a guest of wind. We also generate wind in virtual scenes, enabling users to experience a full sensory experience.

2 Design

2.1 Virtual Reality (VR) Subsystem

2.1.1 Design Description

Based on Unity, this subsystem is responsible for creating virtual scenes for users. The virtual reality (VR) software enables us to build engaging scenarios, tell users interactive stories, and transport people to new worlds by building virtual reality experiences. To be more specific, this module takes sensory information from virtual reality (VR) hardware (includes users' location, head position information, and physical actions) and external perception module (includes wind direction and wind speed) as input, and output is constructed virtual scenes based on the data. If there is an odor source in the scene, virtual reality (VR) software will also output the head position and users' position towards the odor source to the scent simulator. This data transmission will be reached by Inter-Process communication protocols using sockets. We design a classic village scene as a whole for all 5 daily scenes to happen, including a writing program, a tea-drinking scenario, an apple-picking program, a farmland, and a garden. The corresponding scents the user expected to smell while engaging with the virtual world are ink, tea, apple, soil, and flower respectively. The block diagram of this subsystem is shown below.

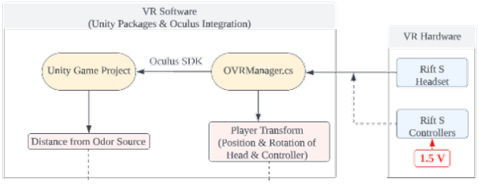

Figure 3: Block Diagram of Virtual Reality (VR) Subsystem

The overview of our built virtual scenario is shown in Figure 4. In the 3D writing program, the user will smell a subtle ink scent as he gets closer to the brush and the intensity of the smell increases when the user starts grabbing and writing. The simulation result is shown in Figure 5. In the tea-drinking scenario, the user can smell the scent of the tea while "drinking" it. The simulation result is shown in Figure 6. In apple plucking scenarios, the user can harvest apples from apple trees and interact with all the apples in the scene. The simulation result is shown in Figure 7.

Figure 4: Overview of Simulation Result

Figure 5: Simulation result for 3D writing

Figure 6: Simulation result for tea drinking

Figure 7: Simulation result for apple garden

In the farmland scenario, the user can walk onto a piece of farmland and get down closer to the ground to smell the scent of fresh soil. The simulation result is shown in Figure 8. Finally, in the garden scenario, the user can smell the scent of flowers in the garden. The effect of the external perception module is implemented in this scenario by involving the data from the wind sensor in the real world in this simpler scenario. The simulation result is shown in Figure 9.

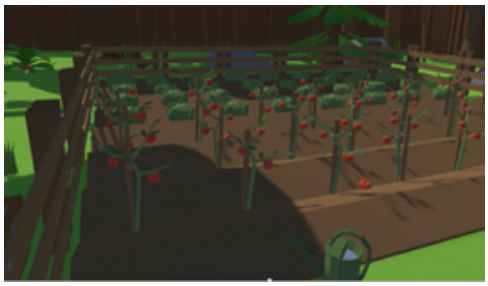

Figure 8: Simulation result for farmland

Figure 9: Simulation result for garden

The virtual reality hardware provides the hardware basis for virtual reality (VR) software and is responsible for motion capture and model rendering, which uses Rift S as the key hardware component. It consists of a virtual reality (VR) headset and two controllers. Through the headset, the user (as the camera inside the project) can see virtual scenes. The headset can track the direction of users' sight. The Rift S uses an LCD with a resolution of $1,280 \times 1,440$ per eye, and refresh rates are slightly lower at $80Hz$ to $90Hz$. The controllers are used to track the position of the hands and control the experience inside the virtual

reality (VR) environment. They feature a protruding black handle and a ring extending from the top for six degrees of freedom (6DOF) position tracking via a camera mounted on the headset. The rings extend to the physical controls of each device and wrap around the user's thumb while giving them plenty of room to move. The data collected by the sensors will be transmitted to the virtual reality (VR) software subsystem for virtual scene generation and movement information pre-processing.

2.1.2 Design Justification

To make sure that the scent is generated synchronously, this distance is calculated at the updating frequency of the VR rendering, which is set to be 0.02 seconds. Then this distance is sent to the scent simulator through UDP. However, to be compatible with the Arduino data transmission efficiency, the sending frequency is set to be 6 seconds. This is accomplished by calling InvokeRepeating("MethodName", initialDelay, repeatRate), which allows you to schedule the execution of a method repeatedly with a specified delay and repeat rate. Notice that the script should be attached exactly to the gameobject with a mesh renderer and a transform instead of a rect transform to guarantee the performance.

2.2 Scent Simulator Software

2.2.1 Design Description

For our project, we need to relate the emission rate in the real world with the user's distance from various scent-emitting sources in the virtual world. The block diagram of this subsystem is shown below in Figure 10.

Let Q_1, P_1 denote the scent emission rate and scent concentration at the user's location in the virtual world. Let Q_2, P_2 denote the scent emission rate and scent concentration at the user's nose position in the real world. Our goal is to calculate Q_2 based on the condition $P_1 = P_2$:

$$P_1(r_1, t) = \frac{Q_1}{(4\pi Dt)^{\frac{3}{2}}} \cdot e^{-\frac{r_1^2}{4Dt}} = \frac{Q_2}{(4\pi Dt)^{\frac{3}{2}}} \cdot e^{-\frac{r_2^2}{4Dt}} = P_2(r_2, t)$$

$$Q_2 \propto e^{-\frac{r_1^2 - r_2^2}{4Dt}} \approx e^{-\frac{r_1^2}{4Dt}}$$

The last approximation step is because we are assuming the distance between the user's nose and our scent emitter is negligible compared to the user's distance to the scent-emitting source in the virtual world. This derivation shows that the scent emission rate in the real world is also exponentially related to the distance in the virtual world. Based on this equation, and with our empirical knowledge about the relationships between distance and concentration, we can calculate the desired emission rate in the real world for a given distance in the virtual world.

The determination of the emission rate, as related to the distance within the virtual world, has now been established. The subsequent question is how this emission rate can be regulated. To address this, we introduced a specific simplification or assumption: the concept

of temporal partitioning, wherein time is divided into discrete slots. This arrangement allows for selective activation of a portion of the scent emitter at any specific time slot, as depicted in Figure 11. Each scent molecule is thus associated with a time period that is a multiple of a designated unit time slot. This strategic temporal partitioning provides a mechanism to manipulate the frequency of release for individual scent molecules.

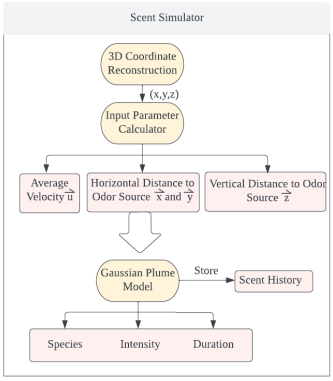

Figure 10: Block Diagram

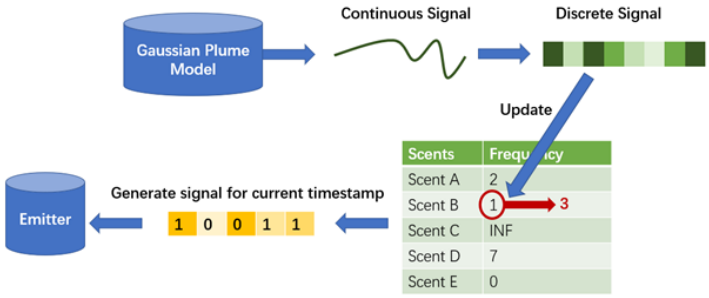

Figure 11: Control of Emission Rate

2.2.2 Alternative Design

To accurately calculate the scent concentration at the user's position in the virtual environment, we derived the Gaussian Plume Model equations that account for factors such as wind direction, wind speed, scent source location, and atmospheric stability class. This involved researching the Gaussian Plume Model and adapting it for our specific use case.

The Gaussian Plume Model is a widely used mathematical model for predicting the dispersion of pollutants in the atmosphere [1]. It is based on the assumption that the pollutant concentration distribution in the horizontal and vertical planes follows a Gaussian distribution. The equation for the Gaussian Plume Model adapted for our scent diffusion application is given by:

$$C(x,y,z) = \frac{Q}{2\pi\sigma_y\sigma_z U}\exp\left(-\frac{y^2}{2\sigma_y^2}\right)\left[\exp\left(-\frac{(z-H)^2}{2\sigma_z^2}\right)+\exp\left(-\frac{(z+H)^2}{2\sigma_z^2}\right)\right], \quad (1)$$

where:

- $C(x,y,z)$ is the scent particle concentration at the point (x,y,z).
- Q is the scent particle emission rate.
- U is the wind speed.
- H is the height of the scent particle source.

- σ_y and σ_z are the horizontal and vertical dispersion coefficients, respectively.

The dispersion coefficients σ_y and σ_z depend on the atmospheric stability class and downwind distance x. They can be calculated using empirical formulas, such as the Pasquill-Gifford-Turner (PGT) curves [1]–[3]. These formulas provide values of σ_y and σ_z for different stability classes and downwind distances.

The Gaussian Plume Model-based equation serves a dual purpose in our scent simulation module: Firstly, it enables us to predict scent dispersion within the virtual environment, a crucial aspect for accurately determining the scent concentration at the user's nose position. Secondly, it facilitates the conversion of the calculated scent concentration at the user's location into appropriate intensity and duration settings for the scent-emitting device. The specifics of these relationships, as well as their implications for our scent simulation module, will be elaborated upon in subsequent sections. The code implementation details of the Gaussian Plume Model are shown in Appendix A.

Another way to relate scent diffusion in the virtual world and in the real world is to solve the 3D diffusion equation. The 3D diffusion equation describes the propagation of a diffusing substance in a three-dimensional space, considering factors such as concentration, diffusion coefficient, and time. The details of this alternative method are shown in Appendix B.

2.2.3 Design Justification

For Gaussian Plume Model, one sample figure is shown below in Figure 12, showing the scent diffusion in the x-y plane.

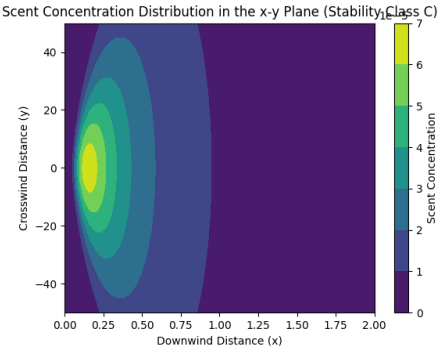

Figure 12: A Sample Simulation Result

The Gaussian Plume Model takes into consideration the wind direction and speed. But it cannot be directly applied to our problem of estimating the scent concentration at the user's location since the Gaussian Plume Model is designed to investigate the macroscopic diffusion of pollutants from a chimney. In comparison, our application is relatively small-scale.

The diffusion equation is suitable for our application but it involves constants that cannot be easily measured. Specifically, the diffusion coefficient D depends on numerous factors such as the nature and state of the substance undergoing diffusion, the medium in which it is diffusing, the temperature and pressure conditions, and the presence of other interacting species. The perfumes used in our system involve various molecules whose sizes cannot be easily estimated. Therefore, we cannot directly apply the solution to the diffusion equation.

However, both the Gaussian Plume Model and the Solution to the Diffusion Equation indicate that the concentration at steady-state is exponentially proportional to the square of the distance from the scent-emitting source, specifically:

$$u(r)_{ss} = A \cdot QC^{-r^2} \tag{2}$$

where:

- $u(r)_{ss}$ is the scent particle concentration at distance r from source in the steady state.
- Q is the scent particle emission rate.
- A, C are constants yet to be determined.
- r is the distance from the scent-emitting source.

Comprehensive tests have been devised to evaluate the module's performance in various scenarios and under different conditions. Throughout our development of the system, the following system-wide tests have also been carried out:

- **User Experience**: This test will involve real users experiencing the "Augmented VR with Smell" system and providing feedback on the scent simulation module's performance. The primary objective is to evaluate user satisfaction and identify areas for improvement.
- **Robustness and Reliability**: This test will focus on assessing the scent simulation module's ability to handle extreme or unexpected conditions, such as abrupt changes in wind direction or speed, or malfunctioning sensors. The goal is to ensure that the module remains functional and accurate under all circumstances.
- **Integration**: Once all subsystems are complete, an integration test will be performed to verify the seamless operation of the entire "Augmented VR with Smell" system. This test will ensure that all subsystems interact as intended and that the overall system functions according to its specifications.

2.3 Scent Emitter

2.3.1 Design Description

The scent emitter subsystem is responsible for emitting scents based on data from scent simulation software. It includes a scent cartridge, an Arduino module, a selection unit,

and an atomization unit with multiple ceramic ultrasonic oscillators. The hardware is designed to be portable and lightweight. The scent cartridges contain individual units with different scents, where a sponge is used to fully absorb the liquid. The Arduino module takes input from the scent simulator regarding the scent species, intensity, and duration, and generates control signals to the selection unit and atomization unit. The selection unit functions as a multiplexer and is connected to five different ceramic ultrasonic oscillators. Based on the selected scent species, the corresponding ceramic ultrasonic oscillator is connected to the rest of the atomization unit to emit the scent. The block diagram of this part is shown above in Figure 13.

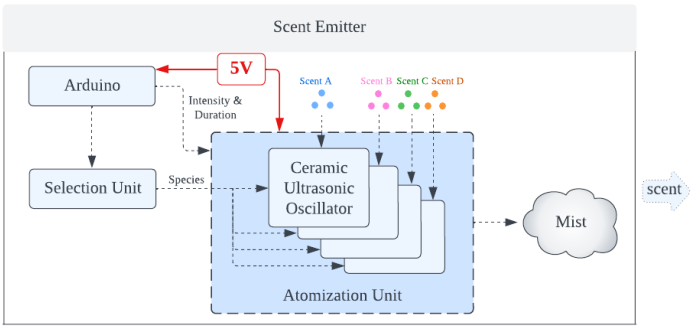

Figure 13: Block Diagram for Scent Emitter

As we will have five different scents, we design five horizontally arranged air outlets. The physical diagram is shown below in Figure 14 and Figure 15.

Figure 14: Physical Diagram of Scent Emitter

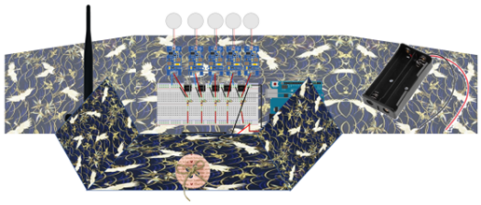

Figure 15: Internal Layout of Scent Emitter

The oscillators used for forming the mist and the control unit used for triggering the oscillators are both put in a neck bag, and the bad is also designed to be worn around the neck. Inside the bag, we put the Arduino and the printed circuit board (PCB) of the atomization unit on the button, which will be separated from the above scent cartridges via a waterproof and corrosion-resistant groove. On top of this control unit, we have five separate scent cartridges. In each scent cartridge, we have a sponge immersed in the perfume and a ceramic ultrasonic oscillator that touches the surface of the sponge. On the top of the box, there are air outlets, and the formed mist will be scattered to the atmosphere from this interface.

As shown in Figure 16, to transmit information wirelessly, we should use two Arduino boards and a wireless transmitter and receiver. This connection will be discussed in the below sections. The circuit can emit scent after receiving the signal from the Arduino board. We use the 8th,7th,6th,5th, and 4th ports on the Arduino board to control scents.

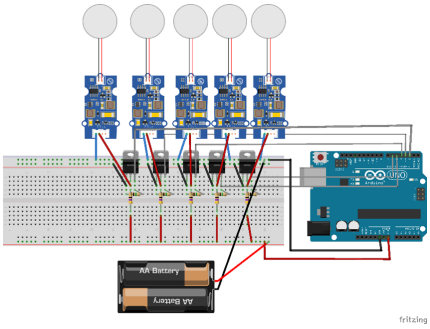

Figure 16: Schematic Diagram

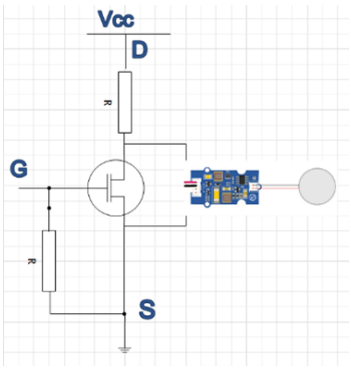

Figure 17: Selection Part

However, they can't power the oscillation circuit directly, because four of these ports can only output 3.3v, it's too low for the oscillation circuit. As shown in Figure 17, the 8th port can output 5v, which is enough for the circuit, but we just use this to test the Bluetooth transmitter and receiver. As mentioned above, there are five identical oscillating circuits in parallel. Figure 4 shows the selection part controlling whether the corresponding circuit will work. The most important component is MOSFET. As for simple MOSFET, if the G gate is high, the D gate and S gate will connect. In our project, we add some improvements. We add a resistor at D. So, if the G gate is high, the current just flows through the resistor. In other words, the oscillation circuit is shorted. When the G gate is low, there is current flow through the oscillation circuit. And because there is a small capacitance designed in MOSFET, we should add a large resistance between the G gate and the S gate. The g gate is connected to the port of Arduino. Figure 17 shows one branch. The other four branches are the same.

The resistance at D gate is hard to choose. If it is large, when the oscillation circuit work, the circuit is too small to work. If the resistance value is small, when the oscillation circuit doesn't work, the resistance would burn out. Therefore, we use resistors with small values and high power. It is 50 Ohms, 5w. And the resistance between the G gate and the S gate should be large enough. Its function is to discharge the internal capacitor. We choose 3.9M Ohms. Resistors that are too large or too small will produce bad results.

For the atomization circuit, the micro atomizing plate is composed of a piezoelectric ceramic ring and a metal steel sheet. The circuit schematic is shown below in Appendix D. By driving the circuit board output PMW pulse width modulation, the piezoelectric ceramic produces hundreds of thousands of times per second high-frequency co-vibration, driving the metal sheet vibration. The liquid ejects from the metal steel sheet of thousands of micro-holes, forms 4-6 micron small molecules, and the liquid molecular structure then

scatter to form the water mist. We connect a 5 V voltage micro atomizing plate. Compared with the traditional voltage requirements of 12 V or 24 V, this micro atomizing plate has the advantage of low energy consumption, which then reduces the requirements for electricity and further expands the use of the product scenario. We design to connect the ceramic ultrasonic oscillator's one pin to Arduino's one IO, which allows the oscillator to work during a period of time when a digital signal is in a logic high state, and stop to oscillate otherwise.

2.3.2 Alternative Design

There are several options for the external structure. A simple and straightforward design is to place the emitter on the ground or on the table. This kind of design doesn't need to worry about stability and don't have strict requirement for weight. However, as the user may move around in the virtual world, the head position and the relative distance towards the emitter are uncertain, and the actual performance will surely be influenced. Hence, we consider designing a wearable device. For debug purposes, the structure cannot be strictly closed. Therefore, we design a pencil-box-like structure, where we can open it easily to debug the circuit and check the connections.

2.3.3 Design Justification

We designed various tests to justify our system.

- Test oscillation circuits: Supply 5v DC voltage to oscillation circuits separately. Test if they can emit scent as soon as the power is supplied.

  Result: All oscillation circuits work well under 5v DC voltage.

- Test whether the amount of scent emitted is related to the voltage: turn the magnitude of voltage from 3v to 6v, observing scent emitting.

  Result: When the voltage is 3v, the LED on the circuit is blinking, this means the voltage is too low. There is no scent emitted. As the voltage improves, more scent is emitted per unit of time. So the amount of scent and voltage are positively correlated.

- Add a selection part to the electronic control part of the scent emitter to choose scents. Using MOSFET to control.

  Result: After changing several schemes and resistors, the correct scent is emitted according to the signals input.

- Test how to adjust scent intensity and duration. There should be at least two emitting speeds.

  Result: The scent emitting speed can't be adjusted by adjusting the voltage because the voltage is constant. Therefore, use emitting frequency to express intensity. It can emit scent continuously or intermittently.

2.4 External Perception

2.4.1 Design Description

The external perception subsystem consists of an abstract space perception module and wind sensors. The abstract space perception module is based on virtual reality hardware, and to be specific, a camera mounted on the headset. The camera captures the room size and worktable height information and sends it to virtual reality (VR) software via DisplayPort 1.2 and USB 3.0 physically. Unity also provides libraries and APIs for users to directly extract the information. The wind sensors are integrated with an external plug-in. The block diagram with physical labels is shown in Figure 18.

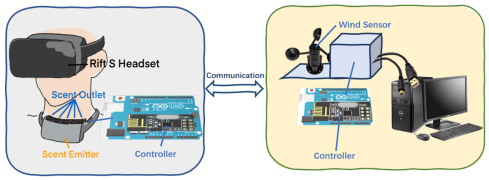

Figure 18: Block Diagram

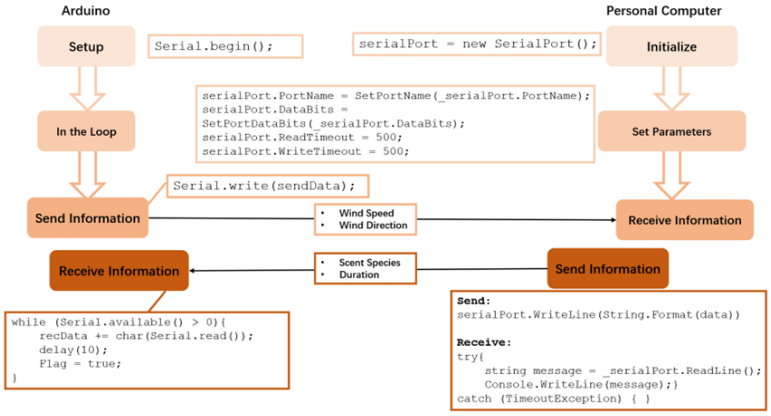

Figure 19: Core Code

On Arduino Uno, pins 0 and 1 are used for communication with the computer. To use these serial ports to communicate with the personal computer, I use a USB-to-serial adaptor. We need to create a bi-direction channel. On the one hand, as shown in the above physical diagram, the wind sensor collects wind information and sends it to the personal computer via a serial port in Arduino and utilizes serial class in C# to get the information. On the other hand, to let scent-simulator software send data to the scent-emitter, the data need to first be transferred to Arduino via serial port, and then transfer to the receiver side Arduino to enable wireless control. According to the official document provided by Arduino [4] and Microsoft [5], for the communication from computer to Arduino, on the Arduino side, I first write a while loop using Serial.available() to detect serial cache. On the C# side, I first create a new SerialPort object using the new $SerialPort()$. Then, I set read/write timeouts, port names, and other appropriate properties. Finally, I use $SerialPort.Open()$ to open the channel and use $serialPort.WriteLine()$ to send correct data. For the communication from Arduino to the computer, the overall logic is similar, and I call different functions like $serialPort.ReadLine()$ for C# and Serial.write() for Arduino. The core code with flowchart is shown in Figure 19. The connection between the microcontroller and wind sensor is quite straight forward as shown in Figure 20. The positive pin from the wind sensor is connected to an Analog-to-Digital Converter (ADC),

which is a series of interfaces on the microcontroller. Analog-to-Digital Converter (ADC) is able to convert continuously varying analog signals into discrete digital signals, which can be processed by digital circuits. The negative pin from the wind sensor is connected to the GND pin on Arduino.

Figure 20: Abstract Connection

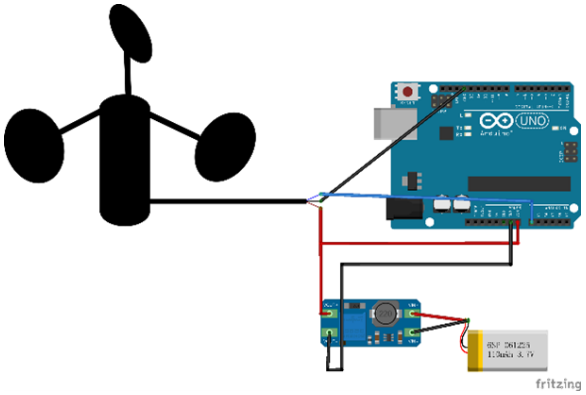

Figure 21: Alternative Design

2.4.2 Alternative Design

There are two kinds of wind sensors in the market, one needs voltage control while another doesn't need. The wind sensor I mentioned in the previous section doesn't need external voltage input. Another wind sensor works under 7-24V DC, so it is obvious that the voltage provided by Arduino is not enough. If we adopt it, we need tp use the MT3608 DC-DC voltage-up converter module to increase the voltage from 3.7V to 7.5V. The output pins of the voltage transformer are connected to the wind sensor's VCC pin and Arduino's Vin Pin. The wind sensor's output is connected to Arduino's ADC interface (A0-A6), I connect to A0 here. I use analogRead(A0) to read data. The schematic diagram for this design is shown in Figure 21.

2.4.3 Design Justification

We did 10000 experiments on the data transmission between the wind sensor and the microcontroller. As the results show, among 10000 experiments, almost all transmissions can satisfy our time constraint. Therefore, it is safe to think that our device meets the high-level requirement we made in previous sections.

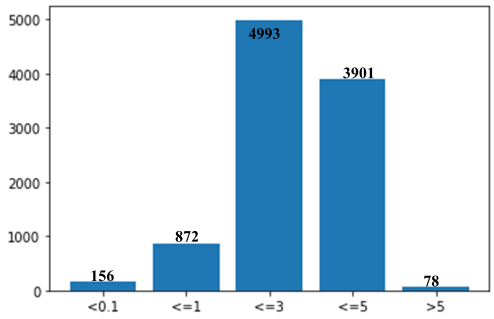

Figure 22: Data transmission time between wind sensors and PC.

2.5 Between Subsystems: Data Transmission

2.5.1 Design Description

Data transmission is a necessary part of software design. Virtual reality (VR) software needs to transmit the user's position information, including distance to the odor source in virtual scenes and the user's head orientation to the scent simulator. To be abstract, scent simulator and virtual reality (VR) software are two processes run on the computer. Overall, the criteria for this part of the design is that the data should be transmitted in real time with high efficiency. Missing some packets in the transmission is acceptable in our scenario, as the virtual reality (VR) software is continuously sending data to the simulator.

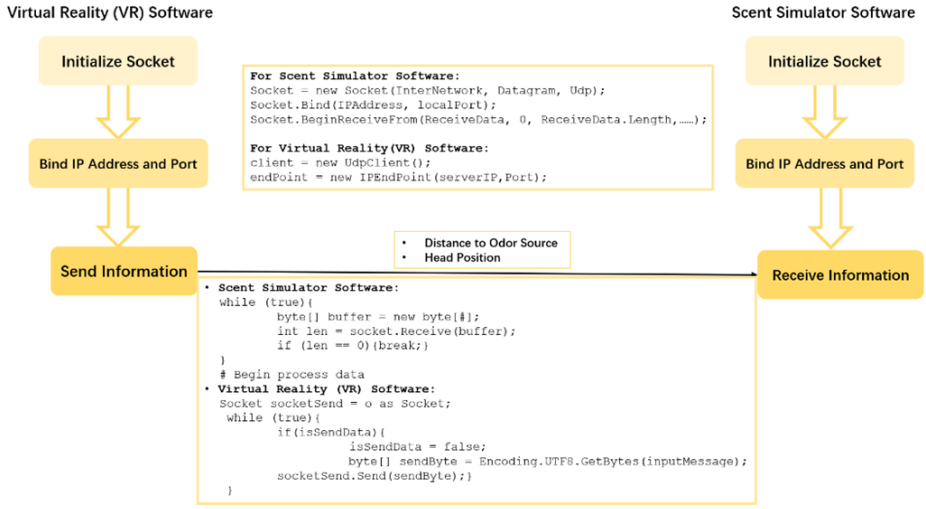

Figure 23: Pseudo-code with flow-chart for data transmission protocol based on sockets.

A socket is an inter-process communication method that encapsulates the details of network communication and allows people to connect and transfer data to any communication endpoint on the network. Actually, we don't require two processes to run on the

same host, as the socket is able to maintain communication between processes on different hosts. We adopt a lightweight data transport protocol that works on top of IP–user datagram protocol (UDP) to ensure transmission efficiency, as packet loss is acceptable in our case. First, both the virtual reality (VR) software and the scent simulator create a socket. They will then call socket API–bind, to bind their IP addresses and ports. After that, virtual reality (VR) software sends data packets to the scent simulator using socket API–$sendto()$, and the scent simulator receives the message using another socket API–$recvfrom()$. The overall pseudo-code with flowchart is shown below in Figure 23.

In general, we use two Arduino micro-controller development boards and two NRF24L01 modules, and each pair works as a transmitter or a receiver. The high-level logical diagram is shown in Figure 24.

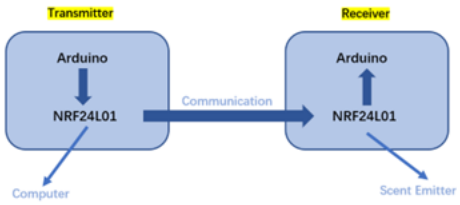

Figure 24: High-level Logical Flowchart for Wireless Data Transmission Method.

NRF24L01 is a single-chip wireless transceiver chip produced by NORDIC that works in the ISM band of 2.4GHz 2.5GHz. Wireless transceivers include frequency generators, enhanced "SchockBurst" mode controllers, power amplifiers, crystal oscillators, modulators, and demodulators. The output power channel selection and protocol settings can be set through the SPI interface. Its greatest advantage is that it has very low current consumption. The schematic diagram and pin definition of this module is shown in Figure 25 and 26.

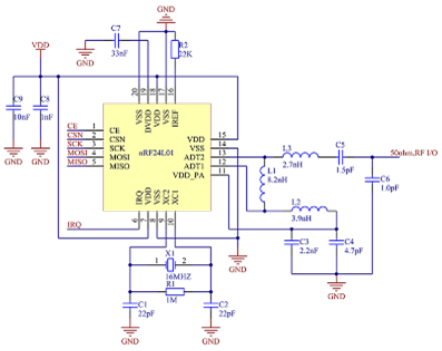

Figure 25: Schematic Diagram of NRF24L01

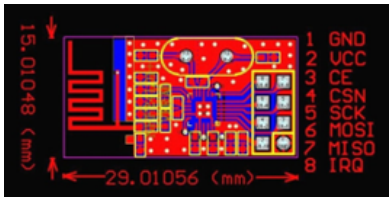

Figure 26: Pin Definition of NRF24L01

By looking up the datasheet, we found that this module works under 1.9V-3.6V. It has 126 communication channels and 6 data channels. Obviously, GND pin and VCC pin should be connected to GND pin on Arduino and 3.3V voltage source respectively. CE pin on

NRF24L01 module connects to pin 8 on Arduino, CSN pin on NRF24L01 module connects to pin 9 on Arduino, SCK pin on NRF24L01 module connects to pin 13 on Arduino, MOSI (MO) pin on NRF24L01 module for output connects to pin 11 on Arduino and MISO (MI) on NRF24L01 module for input connects to pin 12 on Arduino. The pin wiring diagram is shown in Figure 27.

The only thing that remains to be discussed here is programming. To realize the wireless data transmission, I write two pieces of code on the transmitter side and receiver side respectively. Basically, $SPI.h$,$nRF24L01.h$, and RF24.h are three libraries I need. On the receiver side, we first configure NRF24L01 with the correct CE pin and CSN pin. Second, we open the writing pipe and reading pipe with the previously defined address. Then, we start listening. In the loop, we use $radio.available()$ to listen to the incoming data and use $radio.read()$ to read the data. The data then will be processed and control the atomization units. For the transmitter side, the main difference is in the loop, the data received from the scent simulator will be sent using $radio.write()$. The core code with flowchart is shown below in Figure 28.

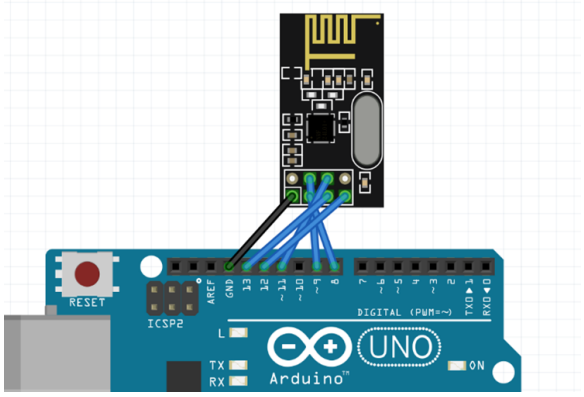

Figure 27: Pin Wiring

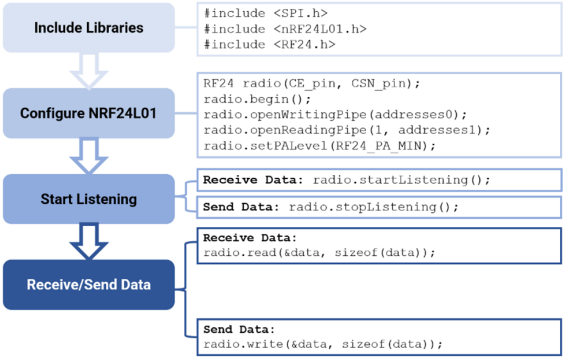

Figure 28: Core Code

2.5.2 Design Alternative

For software data transmission, Based on the official document provided by Unity [6], my alternative considered solution is to use Unity-provided lower-level networking API. This can establish connections, communicate using a variety of "quality of services", enable flow control (like TCP), and do basic statistics. First, we need to set the maximum packet size and the thread timeout limit, and then initialize the network transport layer using $NetworkTransport.Init()$;. As we only need to send data from virtual reality (VR) software to scent simulator software, we think adding one channel is enough. After configuring the channel, we can initialize the host topology and set the maximum connection number. Then, we can bind this network topology with our host IP addresses and an available port. For virtual reality (VR) software, it needs to connect to the scent simulator with the correct IP address and port number, and then send data using provided interface $NetworkTransport.Send()$. For scent simulator software, it receives data by $NetworkTransport.Receive()$. Finally, after communication, we can shut down the con-

nection using $NetworkTransport.Disconnect()$. The overall pseudo-code with flowchart can be found in Appendix C.

2.5.3 Design Justification

We did 10000 experiments on the data transmission between wireless modules and between two software. As the results show, among 10000 experiments, almost all transmissions can satisfy our time constraint. Therefore, it is safe to think that our device meets the high-level requirement we made in previous sections.

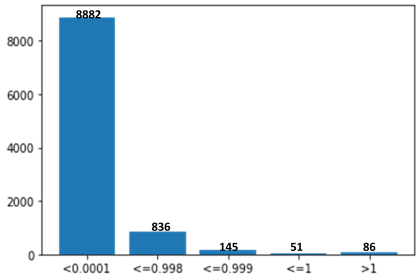

Figure 29: Time between two software

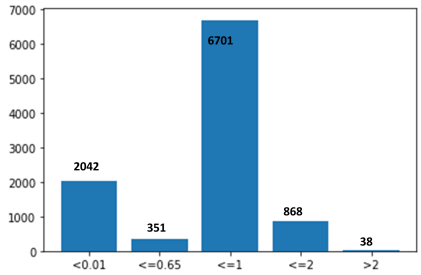

Figure 30: Time between wireless modules

3 Cost and Schedule

3.1 Cost Analysis

3.1.1 Labor

According to the latest available data from UIUC's Engineering Career Services office, the median starting salary for ECE graduates in 2020 was $77,000 per year. Assuming a standard 40-hour workweek, the salary is $37 per hour. For this project, each of us works 12 weeks and 12 hours/week. The total Labor cost is: $4 \times 12wk \times 12hr/wk \times \$37/hr \times 2.5 = \$53280$

3.1.2 Parts

Description	Manufacturer	Part #	Quantity	Cost
Rift S	OCULUS	oculus rift s	1	$597
Ultrasonic atomizer.	MIJIA	MJXFJ01XW	6	$12
Resistor	zave	0805	12	$0.5
Capacitor	Risym	0805	18	$0.7

Triangular inductor	Eckert	6X8(12uH + 440uH)	6	$1
Terminal receptacle.	WXRKDZ	PH2.0	6	$0.25
MOS.	Eckert	SOT 23-3	6	$0.5
SOT 23-6 MCU.	Yingguang	PMS160	6	$0.1
EPGC.	Eckert	FR-4	6	$1
Micro mini 5P.	yeulnelg	/	6	$4.2
Dirt scent perfume.	DORIS FRAGRANCE	2019001299	1	$14
Sweet osmanthus scent perfume.	KEYO NKNK	2021004223	1	$3.5
Tea scent perfume.	KEYO NKNK	/	1	$3.5
Ink scent perfume.	OTHER	J20112481	1	$3
Apple scent perfume.	POACHERMAN	/	1	$4.5

Table 1: Parts List

Total cost: $53280 + $645.75 = $53925.75

3.2 Schedule

Week	Kaiyuan Tan	Yingying Liu	Xiao Wang	Baoyi He
3/13	Discuss with professors; Research physical model.	Discuss with professors; Design hardware structure of scent emitter.	Discuss with professors; Learn how to model virtual structure using Rift S.	Discuss with professors; Design hardware structure of scent emitter.

3/20	Define the physical model of the scent simulator; Use math to theoretically verify the feasibility of the model.	Design circuit of the scent emitter; Decide specific hardware components used in scent emitter; Buy these components.	Continue to learn how to model a virtual world using Rift S; Make a demo virtual scene using Rift S.	Design structures of exterior structures; Compare different materials and decide which materials to use.
3/27	Discuss our virtual scenes; Start writing codes for the scent simulator.	Discuss our virtual scenes; Start building the scent-emitter including hardware and electronic control parts.	Discuss our virtual scenes; Start modeling the virtual world using Rift S.	Discuss our virtual scenes; Decide specific hardware components used in the external perception module.
4/3	Organize calls with our sponsor; Continue to work on modeling the scent simulator.	Build scent-emitter including hardware and electronic control parts.	Continue to work on modeling the virtual world using Rift S; Consider data transmission.	Start building external perception module; Consider data transmission.
4/10	Finish coding part of the scent simulator; Test and debug the software.	Finish prototypes of the scent-emitter; Test and debug the scent-emitter.	Finish modeling the virtual world using Rift S; Test and debug the VR software.	Continue to build the external perception module; Test the reliability.
4/17	Connect scent simulator with external perception model and Rift S; Refine the code.	Connect the scent-emitter with the scent simulator; Test the device and refine based on performance.	Connect Rift S with the scent simulator software; Test and refine the code to reach the best performance.	Finish build the external perception module; Test the connection.
4/24	Assembly subsystems; Test and debug the whole system.	Assembly subsystems; Test and debug the whole system.	Assembly subsystems; Test and debug the whole system.	Assembly subsystems; Test and debug the whole system.

5/1	Prepare mock demo; Write final report draft.	Prepare mock demo; Write final report draft.	Prepare mock demo; Write final report draft.	Prepare mock demo; Write final report draft.
5/15	Fix the broken part	Fix the broken part	Fix the broken part	Fix the broken part
5/22	Work on the final report.	Work on the final report.	Work on the final report.	Work on the final report.

Table 2: Weekly Schedule

4 Requirements and Verification

4.1 Virtual Reality (VR) Software

Requirements	**Verification**	**Results**
1. The demo should run smoothly and consistently with at least 90 Hz frames per second(FPS) to provide visually comfortable and immersive experiences to users.	Use benchmarking tools, such as *Oculus Tray Tool* or *SteamVR Performance Test*, to measure the frame rate of the VR demo while it is running.	The demo run at 85 HZ Frames per second(FPS).
2. The software must be designed to work seamlessly with the latest lineups of Oculus virtual reality (VR) devices.	Ensure that the design is compatible with Oculus' software development kits(SDKs) and tools to guarantee that the software is optimized for Oculus devices and takes advantage of their features and capabilities.	Achieved.
3. The demo needs to include adequate examples to be considered a comprehensive and general implementation of incorporating scent into virtual reality (VR) systems.	The demo should offer at least 3 different scenes/objects, such as a bakery or a bouquet of roses, to incorporate different scents for the users to interact with and experience.	We create 265 objects in total.

4.2 Scent Simulation Software

Requirements	**Verification**	**Results**

1. The simulator should be able to calculate the appropriate intensity and duration of the scents emitting based on the diffusion equation.	Through careful experiment and referring to the Odor Detection Threshold (ODT) which is the minimum concentration of an odorant that can be detected by humans, and Oder Intensity Standard Curves, we can make sure that the scents can be emitted within an acceptable and comfortable range.	Achieved. See 2.2.3 for quantitative results.
2. The real-time result should be delivered correctly to the emitter.	Use Arduino's Universal Serial Asynchronous Receiver Transmitter(USART) protocol to transmit real-time signals to the PC for debugging.	Achieved. Correct and complete message.

4.3 Virtual Reality (VR) Hardware

Requirements	**Verification**	**Results**
1. The virtual reality (VR) hardware should be equipped with a wireless or wired connection with a personal computer (PC) to enable model rendering.	Check whether the personal computer (PC) is equipped with a DisplayPort adaptor. Note that if the connection only supports HDMI personal computers (PCS), Oculus does not guarantee compatibility. So make sure they have a DisplayPort or Mini DisplayPort and a USB 3.0 port.	Achieved.
2. Use at least an Nvidia GTX 1060 or AMD Radeon RX 480 graphics card, an Intel i5-4590 or AMD Ryzen 5 1500X or later CPU, and at least 8GB of RAM for the personal computer (PC).	First, right-click on Computer – Properties and go to the Computer Properties Tab. We can see the basic CPU and memory configurations in the Properties TAB. For more details, we can click Device Manager on the left of the selection. Check these configurations to ensure they satisfy the requirements.	Achieved.

4.4 Scent Emitter

Requirements	Verification	Results
1. The emitter should be driven by an inexpensive, efficient, and user-friendly microcontroller (MCU), enabling easy setup and operation.	The emitter will be streamed by an Arduino Uno R3 as the microcontroller board based on the ATmega328P.	Use Arduino, met requirements.
2. Five ceramic ultrasonic oscillators should all function well when given an appropriate trigger signal.	The range switch of the multimeter should first shift to $2.5V$ DC voltage, and let the red pen connect to the metal sheet, let the black pen be placed horizontally on the ceramic surface, and slight pressure the left hand, and then loosen so that two voltage signals of opposite polarity are generated on the piezoelectric ceramic sheet.	With 5V voltage input and 0.3A current, function normally.
3. The emitter consists of 5 small scent cartridges with scent cartridges installed.	To ensure accurate delivery of scents, we plan to use simulator software to capture the density. If possible, we would like to use specialized equipment, such as gas chromatography-mass spectrometry (GC-MS), to verify the precise chemicals being emitted at each time st.	Achieved.
4. Real-time cues from the API through the OpenXR should be delivered to the Arduino using Web Serial to trigger scents.	Utilize Arduino's USART protocol to send real-time signals to the PC, allowing for comparison with signals received through the API.	Achieved.
5. The power supply must provide a voltage in the range of 4.5-5.5V.	Measure the output voltage using an os- cilloscope, ensuring that the output volt- age stays within the range of 4.5-5.5v.	We use a 5V battery.

4.5 External Perception Subsystem

Requirements	Verification	Results
1. The wind sensor should transmit wind direction and speed to the computer smoothly with a speed of at least 100Kbps.	Send write instructions to the wireless device that has completed pairing, and record the current timestamp. Calculate the writing time based on the current system time and the first timestamp.	Achieved. See 2.4.3 for quantitative results.
2. The wind sensor must be able to detect wind speed in the range of $1m/s$ to $60m/s$ for the resolution ratio to be lower than $0.2m/s$ to fit our requirements.	Design and conduct extensive tests of different wind speeds to make sure the sensor works in these wind conditions. Conduct experiments especially on low wind speed and high wind speed to ensure the property.	Can detect wind speed in 0.75m/s-68.5m/s. With resolution 0.18m/s.
3. The power supply must provide a voltage in the range of 4.5-5.5V for a current load up to 20mA.	Measure the output voltage using an oscilloscope, ensuring that the output voltage stays within the range of 4.5-5.5v.	We use a 5V battery.

5 Conclusion

5.1 Accomplishments

In our senior design project, we successfully augment virtual reality with smell and met all high-level requirements proposed at the beginning of this semester.

Virtual Reality (VR) Software and Hardware Deliver a smooth and immersive virtual reality world. Incorporate diverse scenarios to showcase the integration of olfactory elements in VR scenes.

Scent Simulator Software Employ suitable physical models for scent simulation, incorporating valid modifications and assumptions. Accurately map the relationship between distance and scent intensity for a seamless and natural user experience.

Scent Emitter Select distinct and appealing scents for differentiation. Combinations of different scents should be available. The scent emission device is lightweight, comfortable, and aesthetically pleasing.

External Perception Module he wind speed should affect the virtual scenarios. The wind sensor should transmit wind speed to the computer smoothly with a speed of at least 100Kbps.

Communication between Components Maintain sufficient bandwidth for real-time inter-

component communication. Ensure communication doesn't obstruct user movement or interaction with VR scenes.

5.2 Uncertainties

First, for **portability**, although our device is designed to be light, and did satisfy our high-level requirement of weight, leakage problem still happens sometimes. We require our device to be put on the table or on the ground horizontally, which keeps the liquid from leaking out. By testing, we found that when the angle between the cartridge and the ground is larger than 50 degrees, there is a high possibility for leakage to happen.

Second, for **latency**, although it is safe to think the overall latency satisfies the high-level requirement, there are still glitches that happen. Among 10000 tests for each data transmission experiment, 0.86% failure rate for transmission between software, 0.38% failure rate for wireless modules, and 0.78% failure rate for transmission between wind sensor and personal computer (PC).

5.3 Future Work

First, we can upgrade the wind sensor to incorporate wind direction, and specify wind effect in the scent simulator. Second, as now our device has a large size, we can reduce the size of the scent emitter to make it more portable. For example, we can reduce the size of the bottle which holds perfume. Third, we can import more kinds of odors to enable some more complex virtual reality scenarios. Finally, nowadays, ChatGPT becomes more and more popular. We can let ChatGPT suggests the scent in the VR game. Now we pre-defined the scents for the scenes, but we can let the machine infer that directly. In the future, it can even automatically generate scents based on basic chemical components, which means that chemical reactions could happen inside the scent emitter. However, this definitely includes some more complex and uncertain things related to chemistry, as we know that chemical reaction happens with some condition and will also generate some other signals like heat, which are what wearable devices try hard to avoid.

5.4 Ethical Considerations

Our project has a few potential ethical concerns that must be considered during the design and implementation process. The subsystem and connection schemes involve the danger of leaking private data and ruining the operating system. Concerning the IEEE Code of Ethics, term 1, "to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment" [7], we are careful about potential security bugs for the plug-in that directly connects to the computer, and ensure the attackers cannot steal users' private information via our device. Besides, our proposed device will emit gases, which involve cultural sensitivity. Different cultures have different attitudes towards the scent, and certain scents may be considered offensive or inappropriate in some cultures. Concerning the IEEE

Code of Ethics, term 3, "to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist" [7], we are mindful of these cultural differences and ensure that scent cues are not offensive or inappropriate.

In addition to the above, we pledge to follow the IEEE Ethics guidelines [7] and the ACM Ethics guidelines [8] as closely as possible.

References

[1] D. B. Turner, *Workbook of atmospheric dispersion estimates: an introduction to dispersion modeling*. CRC Press, 1994.

[2] F. Pasquill, "The estimation of the dispersion of windborne material," *Meteorological Magazine*, vol. 90, no. 1063, pp. 33–49, 1961.

[3] F. A. Gifford, "Use of routine meteorological observations for estimating atmospheric dispersion," *Nuclear Safety*, vol. 2, no. 4, pp. 47–57, 1961.

[4] Arduino. ""Serial"." (2023), [Online]. Available: https://www.arduino.cc/reference/en/language/functions/communication/serial/ (visited on 04/05/2023).

[5] Microsoft. ""SerialPort Class"." (2023), [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.io.ports.serialport?redirectedfrom=MSDN&view=dotnet-plat-ext-7.0 (visited on 04/05/2023).

[6] Unity. ""Using the Transport Layer API"." (2020), [Online]. Available: https://docs.unity3d.com/2020.1/Documentation/Manual/UNetUsingTransport.html (visited on 04/04/2023).

[7] IEEE. ""IEEE Code of Ethics"." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 02/08/2020).

[8] ACM. ""ACM Code of Ethics and Professional Conduct"." (2018), [Online]. Available: https://www.acm.org/code-of-ethics (visited on 03/07/2023).

[9] J. Crank, "The mathematics of diffusion," *Oxford University Press*, 1975.

[10] R. Phillips, *Aph162*, Website, Course Materials, 2006. [Online]. Available: http://rpdata.caltech.edu/courses/aph162/2006/Protocols/diffusion.pdf.

Appendix A Code of Scent Simulator

```
import numpy as np
import matplotlib.pyplot as plt

def gaussian_plume_model(Q, x, y, z, U, H, sigma_y, sigma_z):
    C = (Q / (2 * np.pi * sigma_y * sigma_z * U)) * \
        np.exp(-y**2 / (2 * sigma_y**2)) * \
        (np.exp(-(z - H)**2 / (2 * sigma_z**2)) + np.exp(-(z + H)**2 / (2 *
    sigma_z**2)))
    return C

def dispersion_coefficients(stability_class, x):
    coefficients = {
        'A': (213, 0.91, 122, 0.77),
        'B': (156, 0.92, 90.6, 0.73),
        'C': (104, 0.92, 61.0, 0.70),
        'D': (68.0, 0.94, 49.6, 0.65),
        'E': (50.5, 0.95, 34.0, 0.64),
        'F': (34.0, 0.96, 15.9, 0.60)
    }

    a_y, b_y, a_z, b_z = coefficients[stability_class]
    eps = 1e-6  # Small constant to prevent divide by zero error
    sigma_y = a_y * ((x + eps) ** b_y)
    sigma_z = a_z * ((x + eps) ** b_z)
    return sigma_y, sigma_z

# Input parameters
Q = 1.0  # Scent particle emission rate
U = 5.0  # Wind speed
H = 25   # Height of the pollutant source
z = 2    # Vertical distance (fixed)
stability_class = 'C'

# Create a grid of x and y values
x_vals = np.linspace(0, 2, 100)  # Downwind distance values
y_vals = np.linspace(-50, 50, 100)  # Crosswind distance values
X, Y = np.meshgrid(x_vals, y_vals)

# Calculate scent concentration for each point in the grid
C = np.zeros_like(X)
for i in range(X.shape[0]):
    for j in range(X.shape[1]):
        sigma_y, sigma_z = dispersion_coefficients(stability_class, X[i, j])
        C[i, j] = gaussian_plume_model(Q, X[i, j], Y[i, j], z, U, H, sigma_y,
    sigma_z)

# Create a contour plot of the scent concentration distribution
plt.contourf(X, Y, C)
plt.colorbar(label="Scent Concentration")
plt.xlabel("Downwind Distance (x)")
plt.ylabel("Crosswind Distance (y)")
plt.title(f"Scent Concentration Distribution in the x-y Plane (Stability Class
```

```
    {stability_class})")
plt.savefig("gaussian_plume.png")
```

Appendix B Diffusion Equation

By solving the 3D diffusion equation, it becomes possible to simulate the dispersion and spread of scents within the VR environment, allowing for a more realistic representation of smell and enabling the creation of immersive olfactory experiences. The 3D diffusion equation presented in [9] is shown below:

$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \tag{3}$$

In this equation, u is the concentration of the diffusing substance, t is time, x, y, and z are the spatial dimensions, and D is the diffusion coefficient.

Solving this differential equation requires numerous tricks and efforts. Utilizing the method mentioned in [10], we arrive at the following result:

$$u(r,t) = \frac{Q}{(4\pi Dt)^{\frac{3}{2}}} \cdot e^{-\frac{r^2}{4Dt}} \tag{4}$$

where:

- $u(r,t)$ is the scent particle concentration at distance r from source at time t.
- Q is the scent particle emission rate.
- D is the diffusion coefficient.
- r is the distance from the scent-emitting source.

Appendix C Alternative Data Transmission Design

For software data transmission, Based on the official document provided by Unity [6], my alternative considered solution is to use Unity-provided lower-level networking API. This can establish connections, communicate using a variety of "quality of services", enable flow control (like TCP), and do basic statistics. First, we need to set the maximum packet size and the thread timeout limit, and then initialize the network transport layer using $NetworkTransport.Init()$;.

As we only need to send data from virtual reality (VR) software to scent simulator software, we think adding one channel is enough. After configuring the channel, we can initialize the host topology and set the maximum connection number. Then, we can bind this network topology with our host IP addresses and an available port. For virtual reality (VR) software, it needs to connect to the scent simulator with the correct IP address and

port number, and then send data using provided interface $NetworkTransport.Send()$. For scent simulator software, it receives data by $NetworkTransport.Receive()$. Finally, after communication, we can shut down the connection using $NetworkTransport.Disconnect()$. The overall pseudo-code with flowchart is shown below.

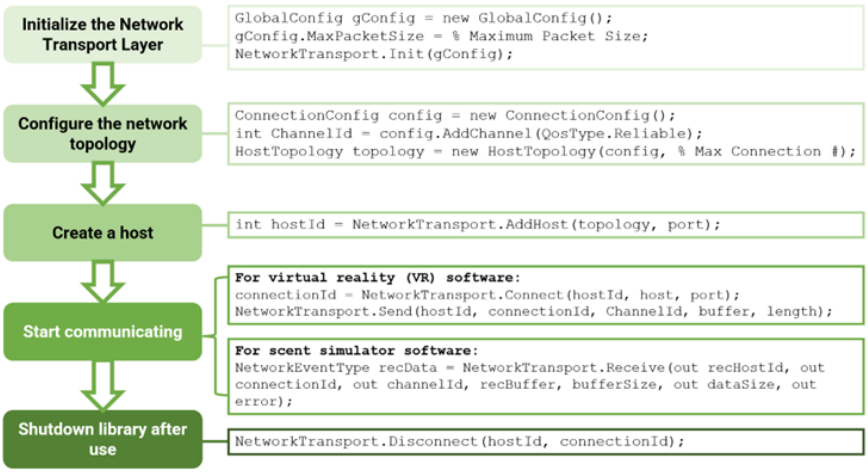

Figure 31: Pseudo-code with flow-chart for data transmission protocol based on Unity provided packages

Appendix D Circuit Schematics of Atomization Unit

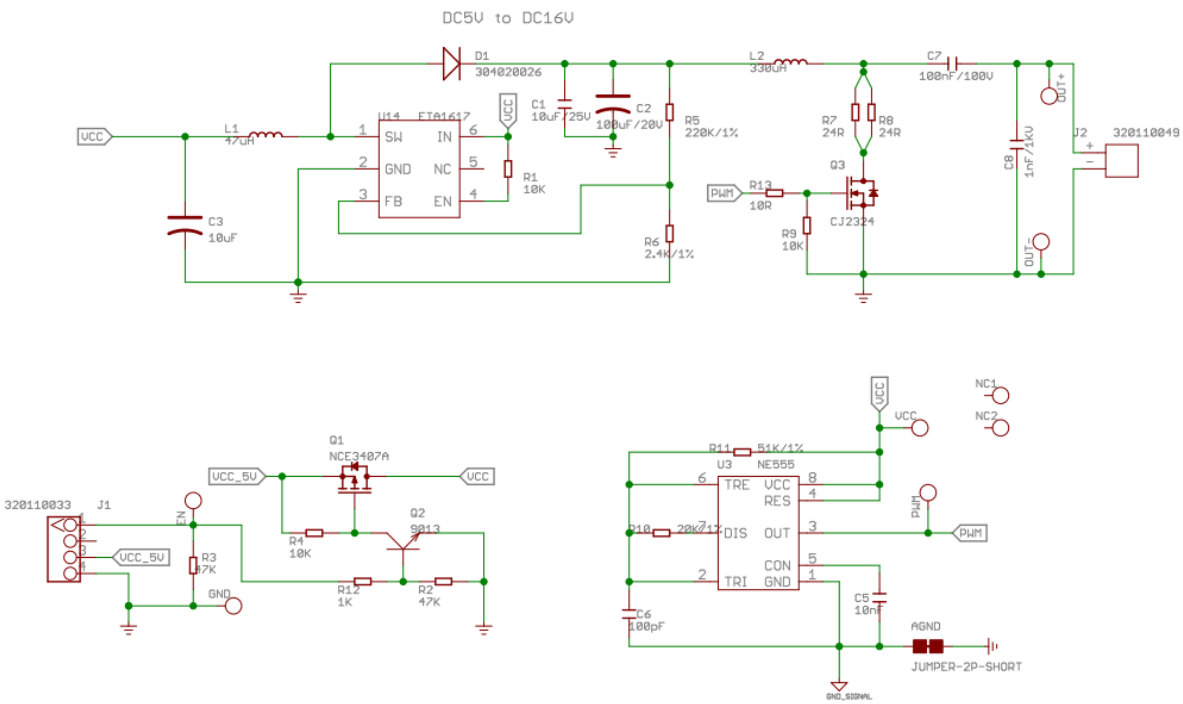

Figure 32: Circuit schematics of Atomization Unit