



**DEPARTMENT OF ELECTRICAL
AND ELECTRONICS
ENGINEERING**

EE491 PROJECT REPORT

Automatic Camera Directing
using Sound Source Localization

Onur TRKMEN - 240206025

Assoc. Prof. Dr. Őevket GMŐTEKİN

DATE: 15/01/2023

ABSTRACT

The aim of this thesis is automatically directing a camera, which is controlled over the VISCA Protocol in an Arduino environment, using sound source localization. Sound source localization is the process of finding the location of a sound source like a clap, and it has a wide range of applications such as surveillance, robotics, military operations to detect a gunshot, etc. In this project, two microphone sensors are used to capture the sound waves, and the time difference of arrival (TDOA) method is employed to estimate the location of the sound source. The estimated location is then used to control a camera, which is connected to an Arduino board, to automatically rotate towards the sound source. The system has been tested in various indoor environments, and the results show that it accurately localizes the sound source. This research also presents a detailed study on the effect of sound waves on the performance of sound source localization, and it provides insights for future work in this field.

TABLE OF CONTENTS

ABBREVIATIONS	3
LIST OF FIGURES	4
1. INTRODUCTION	5
1.1. General Description	5
1.2. Research Motivation	5
2. PROBLEM DEFINITION	6
2.1. Microphone Arrays	6
2.2. The Propagation of Sound	7
2.3. VISCA Protocol	8
3. PROPOSED SOLUTION	9
3.1. Time Difference of Arrival (TDOA)	9
3.2. Communication with the Camera over VISCA using RS-232	12
4. RESULTS AND DISCUSSIONS	14
5. CONCLUSIONS	15
REFERENCES	16

ABBREVIATIONS

2D: Two-dimensional

3D: Three-dimensional

CCTV: Closed-Circuit Television

DOA: Direction of Arrival

DSR: Data Set Ready

DTR: Data Transmission Ready

IDE: Integrated Development Environment

RXD: Receive Data

TDOA: Time Difference of Arrival

TOA: Time of Arrival

TXD: Transmit Data

LIST OF FIGURES

Figure 1	Swedish Antiaircraft Defense System in WWII.....	5
Figure 2	KY-037 SparkFun Microphone Sensor	6
Figure 3	Example of Plane Wavefronts.....	7
Figure 4	VISCA Commands for Absolute and Related Position	8
Figure 5	Visual Explanation of The System	9
Figure 6	Angle the Camera Makes with Sound Source	10
Figure 7	PIN Connections.....	12
Figure 8	Examples of Time Differences and Their Angles as Result	14

1. INTRODUCTION

Localization of a sound source is important in many areas. All of the living creatures in the world have the ability to find the location of a sound source. Thus, they are able to identify any other dangerous creatures and protect themselves. In real-time applications, such as search and rescue missions, and military operations to detect a gunshot, help us understand that the sound source localization plays a big role in people's safety. There are a lot of research and studies that helped me during this project. The popularity of microphone arrays has grown as a result of their capacity for TDOA estimation. The process of sound waves using microphone arrays with time of arrival (TOA), direction of arrival (DOA), TDOA methods involves related works in sound source localization field. For instance, TDOA has been proposed in a study to build an apparatus that could pinpoint the location of a sound source [1]. This method is widely used in the sound source localization field.

1.1 General Description

In this thesis, I have proposed a system for sound source localization that uses two microphone sensors and the TDOA method to find the estimated location of a sound source. The estimated location is used to automatically rotate the camera which can be controlled over VISCA Protocol in Arduino environment. In recent years, by using spatially and equally distributed microphone arrays in both 2D and 3D has increased in order to find the location of the sound source [2]. It is possible to see its applications in the medical field such as providing better quality hearing aids for deaf people.

1.2. Research Motivation

Humankind's interest in sound source localization started between World War I and World War II. In order to find the location of the enemy aircraft, nations were creating antiaircraft defense systems which works with the principle of sound source localization. As it is shown in Figure – 1, Swedish Antiaircraft Defense which is equipped with a microphone array was used to pick up the noise of the aircraft's engines. In this project, my aim and focus was being able to create a system that can find the estimated location of an impact sound source like a clap or a thud. Implementing this system to any surveillance camera would keep people safer.



Figure 1: Swedish Antiaircraft Defense System in WWII [3]

2. PROBLEM DEFINITION

2.1. Microphone Arrays

An array of microphones uses two or more microphone sensors to detect sound waves. In order to detect them properly, there are two factors to be careful about. The two factors of directionality, and sensitivity should be taken into consideration while matching microphone sensors in a microphone array. The directionality of a sensor is the direction that it picks up sounds. Some microphone sensors can only pick up sounds from one direction, and others can pick up sounds from any direction. Sensitivity of microphone sensors is another factor to adjust while setting up a microphone array system. Sensitivity is the gain of a microphone sensor that picks up sound waves. All of the sensors' sensitivity levels should match each other. Otherwise, one microphone sensor might detect a sound, while others do not. That is an unwanted situation and would affect all of our calculations [4]. In the light of these, I have prepared a system with two microphone sensors.

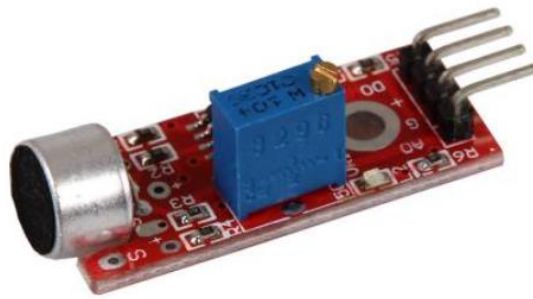


Figure 2: KY-037 SparkFun Microphone Sensor

As can be seen in Figure – 2, I have used SparkFun's Ky-037 model microphone sensors. KY-037 microphone sensors have 4 pinouts. These are +V, GND, Digital Signal, and Analog Signal pinouts. They can't measure or calculate any information about a sound wave's amplitude or frequency. Through digital pinout, it gives only HIGH and LOW values that represent if sound is detected or not. The challenge with them was setting their sensitivity. By adjusting the potentiometer while checking its voltage on a multimeter, I have tried to obtain the same sensitivity level that only detects impact sounds like a clap sound for both sensors. There are better microphone sensors that can even calculate amplitude and frequency of a sound wave, but they are more expensive and harder to find. Therefore, I chose this model to move on with in my project.

2.2. The Propagation of Sound

Sound travels through compressible substances like water or air. Sound waves can be reflected, refracted or attenuated by external factors. As a consequence of these external factors, sound waves do not propagate properly every time they are produced. In this project, by using sound waves, we can find the sound source's location. There are several methods that has been explained in the Introduction part. For my study, I have done my calculations using TDOA in order to find the estimated angle that sound waves make with the microphone sensors. I have assumed that the sound source is producing sound waves that propagate as if they are plane wavefronts in the air. In Figure – 3, there is an example that shows a microphone array with two microphone sensors and how plane wavefronts travel towards the sensors.

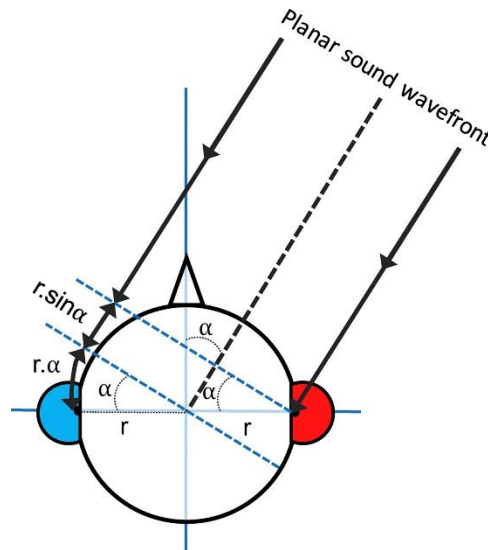


Figure 3: Example of Plane Wavefronts [5]

Since TDOA lets me calculate the difference between sound detection times of two microphone sensors, I can use that time information with combining speed of sound, which is 343 meters per second in the air. Then, velocity equation and trigonometric calculations helped me to find the desired angle.

2.3. VISCA Protocol

The angle between sound waves and microphone sensors will then be used in rotating the camera to the sound source. The camera that I have used in this project is called Sony EVI-D31V. This model uses VISCA Protocol to communicate with Arduino. VISCA Protocol steps in at this point. VISCA Protocol is a professional camera control protocol used with PTZ, which stands for pan-tilt-zoom, cameras. It is designed by Sony, and it's based RS-232 serial communications at 9600 bits per second with one start bit, one stop bit, no parity and no flow control [6]. VISCA Protocol is used to send the command to the camera byte by byte. There are fixed commands to rotate the camera.

Absolute position	8x 01 06 02 VV WW 0Y 0Y 0Y 0Y 0Z 0Z 0Z 0Z FF	YYYY : pan position FC90 to 0370 (center 0000)
Relative position	8x 01 06 03 VV WW 0Y 0Y 0Y 0Y 0Z 0Z 0Z 0Z FF	ZZZZ : tilt position FED4 to 012C (center 0000) W : 0 UpRight, 1 DownLeft

Figure 4: VISCA Commands for Absolute and Related Position [7]

In Figure – 4, there are example commands that are derived from camera's service manual. Since our aim is to rotate the camera to the estimated location of the sound source, I have used the absolute position command.

3. PROPOSED SOLUTION

3.1. Time Difference of Arrival Method (TDOA)

The idea of Time Difference of Arrival method is calculating the time difference between sound detection times of two microphone sensors.

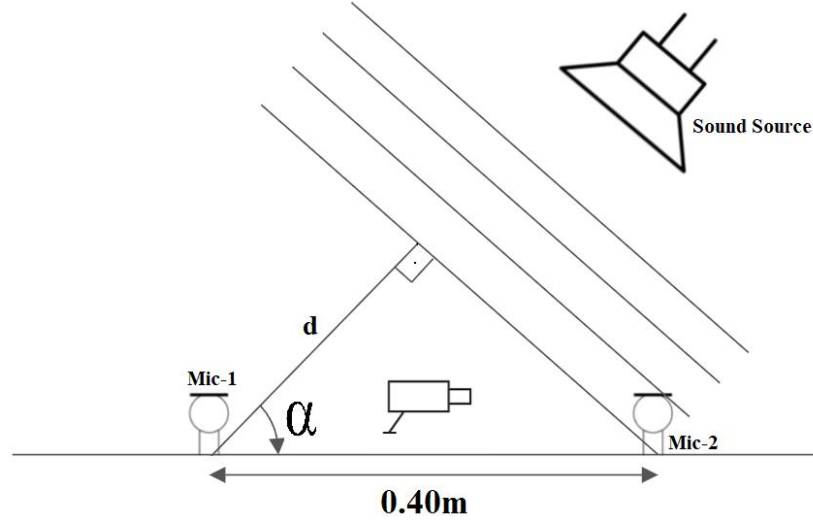


Figure 5: Visual Explanation of The System

As shown in Figure – 5, sound waves propagate from the sound source towards the sensors as if they are plane wavefronts. When microphone sensor – 2, which is closer to the sound source in this case, detects the sound, a microsecond timer starts counting the time in Arduino IDE. Right after the microphone sensor – 1 also detects the sound, its microsecond timer also starts counting. Then, the difference between the values that are saved in those timers is the time where the sound has travelled the distance ‘d’. Since we know that the speed of sound is approximately 343 m/s in the air, it’s possible to calculate the distance ‘d’.

$$d = t_d * v_s \quad (1)$$

By using equation 1, which is called velocity equation, I have calculated distance ‘d’ whenever both sensors detect a sound. After that, I needed to find the angle α . In this step, trigonometric calculations had to be done in that right-angled triangle.

$$d = (t_2 - t_1) * 0.000001 * 343.0m/s \quad (2)$$

$$\beta = \cos^{-1}\left(\frac{d}{0.40m}\right) \quad \text{in radians} \quad (3)$$

$$\alpha = \left(\frac{\beta}{2\pi}\right) * 360 \quad \text{in degrees} \quad (4)$$

In equations from 2 to 4, I have found the angle α . In equation 2, calculated time difference value t_d and converted it into seconds. Then, multiplying it with speed of sound gives us the distance 'd'. Since distance between microphone sensors is determined as 0.40m, putting both of the values into arccosine function as in equation 3 gives as the value β , which is in radians. Following, I have converted it to degrees using equation 4. However, the camera should rotate to the sound source. Therefore, by applying the geometrical process that is shown in Figure – 6, the angle that the camera has to rotate was calculated.

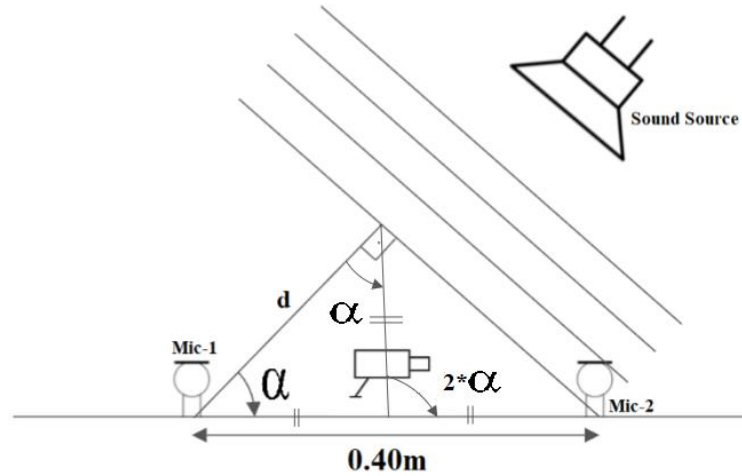


Figure 6: Angle the Camera Makes with Sound Source

While implementing this process in Arduino IDE, I have created two different functions outside of the loop function to find the angle. 'float time_to_radian' returns us the value β and 'float radian_to_degree' returns the value α .

```
float time_to_radian(float time) {
    return (acos((time * 0.000001 * 343.00) / (Distance)));
}
float radian_to_deg(float radian) {
    return ((radian / (2 * PI)) * 360);
}
```

In the loop function, I have controlled the microphone sensors' detection conditions with a couple if statements.

```

if (digitalRead(sensor1) == HIGH) {
    time1 = micros();

    while (digitalRead(sensor2) == LOW) {
    }
    time2 = micros();
    time3 = time2 - time1;

    if (time3 < 1171) {

        radian = time_to_radian(time3);
        degree = radian_to_deg(radian);
        float newdegree = -1 * (180 - 2 * degree);
        Serial.print("Angle: ");
        Serial.println(newdegree);
        SendViscaPanAngle(newdegree);
        delay(20);

        delay(Delay);
    }
} else if (digitalRead(sensor2) == HIGH) {
    time1 = micros();

    while (digitalRead(sensor1) == LOW) {
    }
    time2 = micros();
    time3 = time2 - time1;

    if (time3 < 1171) {

        radian = time_to_radian(time3);
        degree = radian_to_deg(radian);
        float newdegree = 180 - 2 * degree;
        Serial.print("Angle: ");
        Serial.println(newdegree);
        SendViscaPanAngle(newdegree);
        delay(20);

        delay(Delay);
    }
}

```

In these lines of my Arduino code, TDOA method has been applied to the system for two different conditions. As can be seen, there is a limit determined as $1171\mu s$ in order to get into an if statement. The reason behind that limit is that time difference cannot be higher than $1162\mu s$. Since the longest distance that sound can travel after getting detected by one of the microphone sensors is $0.40m$, the time difference gets its maximum value while travelling that distance and it is calculated as approximately $1166\mu s$ using equation 2. If the time difference result gets obtained higher than its maximum value, the system would fail.

3.2. Communication with the Camera over VISCA Protocol using RS-232

Since I have found the angle that camera needs to rotate to, the next step is to establish the connection and send the commands to the camera over VISCA in Arduino IDE. RS-232 is a widely used standard for serial communication transmission data. In this thesis, I have used an RS-232 to TTL converter to make connection between the camera and my Arduino board.

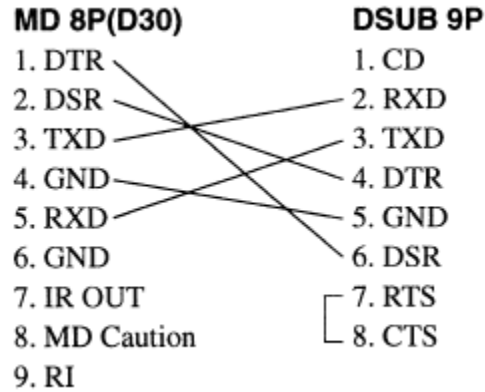


Figure 7: PIN Connections [7]

VISCA IN pinout of the camera is an 8-pin DIN socket. However, RS-232 to TTL converters require a DB9 socket to connect to Arduino. Therefore, I should have soldered the pins of an 8-pin DIN socket to DB9 socket's pins. In Figure – 7, the configuration of PIN connections are shown. 8-pin DIN socket's DTR goes to DB9 socket's DSR, TXD goes to RXD, and vice versa. It means that in order to communicate with the camera, those pins should be cross connected to each other.

When all the cable connections and soldering are done, the angle that is found in the previous steps had to be converted to hexadecimal form in order to put that information in VISCA commands. According to Sony's Service Manual, in order to rotate the camera to the right end side Y values that are in Absolute Position command should be 0370 respectively. 0370 in hexadecimal form is equal to 880 in decimal form. Since our camera can rotate up to 100 degrees, it means that to rotate the camera by one degree, I needed to increase that decimal value by 8.8. For the left side pan movement of the camera, the interval from center to the left end side is 880 in decimal form again. However, I had to subtract the desired decimal value from FFFF in hexadecimal form, which is equal to 65535 in decimal form.

```

byte viscaPanTiltCmd[15]= { 0x81,0x01, 0x06,0x02,
speed, 0x07,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00,
0x00,0xFF };

void SendViscaPanAngle(int ang) {
    word angw = 0;
    word angw1 = 0;

    if (ang >= 0) {
        angw = ang * 8.8;
        angw1 = angw;
        viscaPanTiltCmd[9] = angw1 & 15;
        angw1 = angw;
        viscaPanTiltCmd[8] = (angw1 >> 4) & 15;
        angw1 = angw;
        viscaPanTiltCmd[7] = (angw1 >> 8) & 15;
        angw1 = angw;
        viscaPanTiltCmd[6] = (angw1 >> 12) & 15;
    }
}

else {
    {
        angw = ang * 8.8;
        angw = 65535 + angw;
        angw1 = angw;
        viscaPanTiltCmd[9] = angw1 & 15;
        angw1 = angw;
        viscaPanTiltCmd[8] = (angw1 >> 4) & 15;
        angw1 = angw;
        viscaPanTiltCmd[7] = (angw1 >> 8) & 15;
        angw1 = angw;
        viscaPanTiltCmd[6] = (angw1 >> 12) & 15;
    }
}

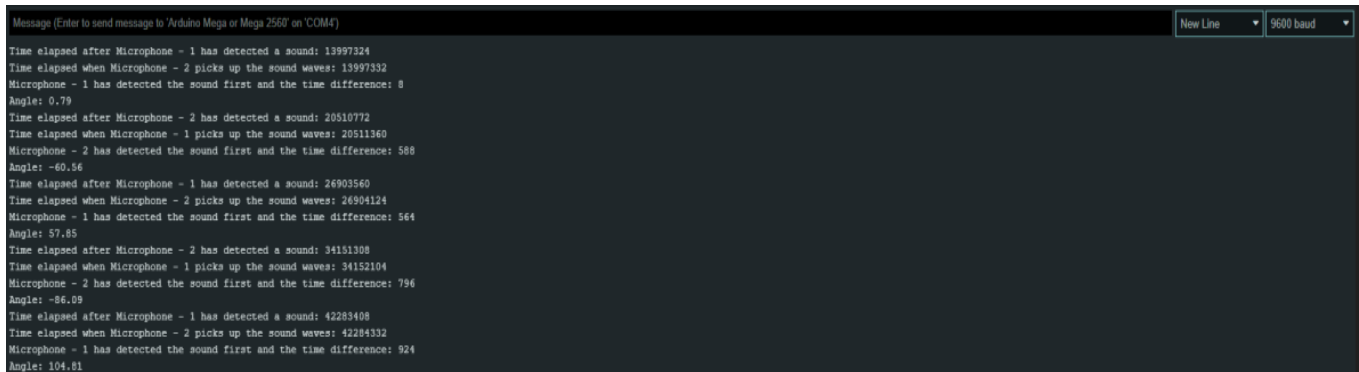
for (int c = 0; c < sizeof(viscaPanTiltCmd); c++) {
    Serial2.write(viscaPanTiltCmd[c]);
    delay(10);
}
}

```

In these lines of the code, my aim was to shift the desired angle value byte by byte and put them into byte array, which is called viscaPanTiltCmd, respectively. Afterwards, I've sent those bytes to the camera.

4. RESULTS AND DISCUSSIONS

Since it's not possible to demonstrate my project in this section, I will show some results in terms of time difference values and angles. I have created 5 different impact sounds to see if the TDOA method works properly and whether our calculations are correct or not.



```
Message (Enter to send message to Arduino Mega or Mega 2560 on COM4)
Time elapsed after Microphone - 1 has detected a sound: 13997324
Time elapsed when Microphone - 2 picks up the sound waves: 13997332
Microphone - 1 has detected the sound first and the time difference: 8
Angle: 0.79
Time elapsed after Microphone - 2 has detected a sound: 20510772
Time elapsed when Microphone - 1 picks up the sound waves: 20511360
Microphone - 2 has detected the sound first and the time difference: 588
Angle: -60.56
Time elapsed after Microphone - 1 has detected a sound: 26903540
Time elapsed when Microphone - 2 picks up the sound waves: 26904124
Microphone - 1 has detected the sound first and the time difference: 564
Angle: 57.85
Time elapsed after Microphone - 2 has detected a sound: 34151308
Time elapsed when Microphone - 1 picks up the sound waves: 34152104
Microphone - 2 has detected the sound first and the time difference: 796
Angle: -86.09
Time elapsed after Microphone - 1 has detected a sound: 42283408
Time elapsed when Microphone - 2 picks up the sound waves: 42284332
Microphone - 1 has detected the sound first and the time difference: 924
Angle: 104.61
```

Figure 8: Examples of Time Differences and Their Angles as Result

As can be seen in Figure – 8, time differences are found by TDOA method. Then, by using the equations 1 – 4, the angles that correspond to the specific time value has been calculated accurately.

In view of these studies, I'm able to find the estimated location of a sound source and rotate the camera towards the source over VISCA Protocol. However, there might be miscalculations during tests. Since sound waves keep reflecting through the surfaces like walls, furniture, etc. microphone sensors may detect one of those reflections and start the timer. Also, KY-037 microphone sensors' characteristics are not capable of detecting sound from far distances. Therefore, the system may not work as desired when the sound source is far away from the sensors. Moreover, Arduino board's microcontroller's clock speed is also an important factor to get correct results. If its clock speed isn't enough to calculate the microseconds over and over, the system would fail again. My microcontroller's clock speed is 16 MHz, which seems enough for this case.

5. CONCLUSIONS

In this paper, the TDOA method was proposed for obtaining the necessary information such as time difference and angle. As the project's main purpose, I have found the sound source's estimated location using that information. Then, by using a camera, which can be controlled over VISCA Protocol in Arduino IDE, we wanted to visualize that the location can be found.

During my studies, I have worked with three different Arduino boards and 5 different types of microphone sensors. Finding the most optimal equipment was one of the challenges in this project. It showed me that the sound source localization requires sufficient equipment to get accurate results. Since I assumed that the sound waves propagate as if they are plane wavefronts, the location that is found is not the exact location. Therefore, as an extension, the exact location and even the sound source's range from the camera can be calculated using more and better microphone sensors. As future efforts, this system can be implemented to any CCTV/Surveillance cameras. In terms of security, a CCTV that rotates to the sound source without any human control would help a lot.

REFERENCES

- [1] Mantle, L., & Soroco, M. (2019). *Direction and Ranging of an Incoming Sound from an Arduino Sound Sensor System*. <https://doi.org/10.14288/1.0398275>
- [2] Thakur, S., & Singh, S. (2022). Sound source localization of harmonic sources in entire 3D space using just 5 acoustic signals. *Applied Acoustics*, 201, 109126. <https://doi.org/10.1016/j.apacoust.2022.109126>
- [3] Wikimedia Foundation. (2022, July 5). *Acoustic location*. Wikipedia. Retrieved January 14, 2023, from https://en.wikipedia.org/wiki/Acoustic_location
- [4] What is a microphone array? (n.d.). Retrieved January 14, 2023, from <http://www.learningaboutelectronics.com/Articles/What-is-an-array-microphone>
- [5] Risoud, M., Hanson, J.-N., Gauvrit, F., Renard, C., Lemesre, P.-E., Bonne, N.-X., & Vincent, C. (2018). Sound source localization. *European Annals of Otorhinolaryngology, Head and Neck Diseases*, 135(4), 259–264. <https://doi.org/10.1016/j.anorl.2018.04.009>
- [6] Wikimedia Foundation. (2022, December 13). *Visca protocol*. Wikipedia. Retrieved January 14, 2023, from https://en.wikipedia.org/wiki/VISCA_Protocol
- [7] Sony. (1999). EVI-D30/D31: Service Manual. Tokyo, JP: Sony Corporation.
- [8] Jia, X., Jia, M., Li, L., & Jia, Y. (2020). Multiple sound source localization by using diffuseness estimation. *2020 7th International Conference on Information Science and Control Engineering (ICISCE)*. <https://doi.org/10.1109/icisce50968.2020.00181>
- [9] Wang, T., & Choy, Y. (2015). An approach for sound sources localization and characterization using array of microphones. *2015 International Conference on Noise and Fluctuations (ICNF)*. <https://doi.org/10.1109/icnf.2015.7288571>