

Relating The Ultrasonic Radar Assignment To Assisted Parking

Ava Delgra and Sophia Hickey

University of Tennessee, Knoxville

505 Summer Place – UTT 946, Knoxville, TN 37902-2807

adelgra@vols.utk.edu

shickey5@vols.utk.edu

Submitted to TRACE for potential publication

Abstract - Assisted parking systems have become increasingly essential to modern vehicles to aid drivers in safely maneuvering into parking spaces. This paper presents the design and implementation of an Arduino-based assisted parking system, intended to measure distances and control an output device based on those measurements. The device utilizes an ultrasonic sensor, which emits sound waves and measures the time taken for the waves to reflect back, allowing for accurate distance calculations. The system is programmed to produce an auditory output depending on the distance calculation. It offers a practical application for proximity-based control in environments where auditory feedback is necessary - such as in a parking situation.

Keywords -

Arduino, Assisted Parking, Baud Rate, Code Functions, Collision Alert System, Electrical Polarity, Input/Output Pin, Loops, Parking Safety, Proximity Detection, Ultrasonic Sensors.

Abbreviations -

BR - Baud Rate
cm - centimeter

VCC - Voltage at Connector

GND - Ground Pin

μs - microseconds

(Within the code: many functions are abbreviated)

I. Introduction

As the use of automotive transportation continues to rise, automotive engineers have recognized the need for systems that assist drivers in safely parking their vehicles. Parking accidents alone account for 20% of car accidents here in the United States [1], and this doesn't even take into account all of the accidents due to cars hitting stationary or moving objects while parking.

Car repairs can be very expensive. Parking lot accidents can range anywhere from a few hundred dollars for some scratch repair or more cosmetic-based fixes, and up to several thousand for things such as replacement parts. [2] This doesn't even take into account the driver's insurance cost increase as just one accident can increase the insurance price by 45% depending on whose fault it is, location - both type of lot and state/city, and records [3]. A driver could also have to pay more money for any damages done to the property, potential court cases, any medical fees, or damage fees for the other vehicle if he or she is at fault.

Beyond the price, the driver could potentially cause people to get seriously hurt. The automotive industry's goal should be to protect people at all costs - both inside and outside of the car. As cars become more and more popular and more accessible to the public, the amount of accidents will continue to rise. In order to combat this, car companies and external companies that focus on parking technologies have come together to try and improve these problems and increase the safety of their drivers. This is such a huge focus not only because it improves their safety ratings which helps save lives, but if they are able to do everything they can within technology limits to protect everyone both inside and outside of the car, and they can promise fewer accidents due to their increased safety, then people will in turn look to those car companies when buying cars which makes their business more profitable.

Overall, due to the major changes in society through the rise of cars and technology, people such as automotive engineers are looking to find ways to make the roads safer and ease some of our major roadway problems through the use of assistive driving and parking technologies.

II. Materials and Methods

The process of effectively creating and executing this project contains two main procedures; writing the Arduino code, and assembling the circuit board. Both processes would be essential to the success of this project.

A. *Writing and Explanation of the Arduino Code*

// ECE 301 Final Project
 // Relating The Ultrasonic Radar Assignment
 To Assisted Parking

```
// Ava Delgra and Sophia Hickey
// Fall 2024

// defines pins numbers
#define IN4 4
const int trigPin = 5;
const int echoPin = 6;
// defines variables
long duration;
int distance = 0;
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the
  trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the
  echoPin as an Input
  pinMode(4, OUTPUT);
  Serial.begin(9600); // Starts the serial
  communication
}
void loop() {
  if ((distance < 50) && (distance > 0))
  {
    digitalWrite(IN4, HIGH); // turn the LED
    on (HIGH is the voltage level)
  }
  else if ((distance > 51) && (distance < 150))
  {
    digitalWrite(IN4, HIGH); // turn the LED
    on (HIGH is the voltage level)
    delay(10*(distance-50));
    digitalWrite(IN4, LOW); // turn the LED
    on (HIGH is the voltage level)
    delay(10*(distance-50)); // wait for a set time
  }
  else if ((distance > 151) && (distance < 200)) {
    digitalWrite(IN4, HIGH); // turn the LED
    on (HIGH is the voltage level)
    delay(1000);
    digitalWrite(IN4, LOW); // turn the LED
    on (HIGH is the voltage level)
    delay(1000);
```

```

}
else if (distance > 200){
  digitalWrite(IN4, LOW);
}
delayMicroseconds(10);
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro
seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave
travel time in microseconds
duration = pulseIn(echoPin, HIGH);
// Calculating the distance
distance = duration*0.034/2;
// Prints the distance on the Serial Monitor
Serial.print("Distance: ");
Serial.println(distance); //distance is the
returned integer, in cm
}

```

This Arduino code is designed to program an ultrasonic sensor to both measure distance and control an output device (in the case of this project, a loud, electronic buzzer) based on the measured distance.

The program begins by first defining the pin numbers for the ultrasonic sensor and the output device. The sensor has two different pins - the “trigPin” and the “echoPin” (pin 6). The trigPin sends an ultrasonic pulse or wave at a frequency above the range of human hearing, to trigger the sensor. The pulse travels until it hits an object (such as a car, wall, or barrier), rebounds back in the direction it came from, and is received by the echoPin of the sensor [4]. The sensor records and measures the time it takes for the pulse to be emitted, hit the object, and return to the echoPin.

Using the following equation, this time is used to calculate the distance to the object.

$$distance = duration * 0.034/2 \quad (1)$$

Where the variable *duration* refers to the time duration and 0.034 is the speed of sound (0.034 cm/μm). Equation (1) is divided by two to account for the fact that the waves travel to the object and back.

The Arduino uses the function “setup()” for two separate purposes. Firstly, the code within the function is used to configure the sensor pins and the output pin, IN4 (pin 4), as an output to control the device. Secondly, the code “Serial.begin(9600)” initializes the serial communication between the Arduino and the computer. The number 9600 refers to the baud rate (BR), or the speed at which data is to be transferred between the Arduino and the computer [5].

In coding, loops are created to allow a block of code to be executed over and over again, a specified, or even infinite, amount of times [6]. In the Arduino code written for this project, the function “loop()” is used to continuously run and check the measured distance. Based on the distance, the code alters the behavior of the output pin.

To create different circumstances when and alter the behavior of the pin, the code uses a collection of if and else if statements. For example, lines 20-24 of the Arduino code reads:

```

void loop() {
  if ((distance < 50) && (distance > 0))
  {
    digitalWrite(IN4, HIGH);
  }
}

```

The first line uses an if statement to define a specific condition in which the following block of code can be executed. It says that if the input

for distance, based on the sensor, falls within the range of greater than zero but less than fifty, then the following function (`digitalWrite`) will be performed. The next else if statements are similar, but define varying distances.

The output pin is used to control an external device - like an LED, or another component, such as a motor or relay. In the case of this project, this external device is the buzzer, which can project a constant, loud buzzing sound. Within the loop, the code uses the function "`digitalWrite()`" to set the output pin to either HIGH or LOW. When the pin is set to HIGH, the code tells the Arduino board to deliver voltage to the connected device, and the buzzer's sound is released, letting the user know that a specific distance between the sensor and the detected object has been reached. When the pin is set to LOW, the voltage supply to the buzzer is cut off, and in turn, the noise stops. Therefore, the code is written so that the behavior of the IN4 pin depends on the measured distance from the object to the ultrasonic sensor, and tells the buzzer to go off when that distance is small and when it decreases.

The code also uses the function "`delay()`" to control the timing. This function is used in the code to control how long the buzzer stays on or off based on the measured distance. For distances between 51 and 150 centimeters, the delay is proportional to the distance - or the buzzer is coded to stay on longer than if the object is farther away. Similarly, the `delay()` function is used in other parts of the code to create a specific timing mechanism for different behaviors of the buzzer, like for example, turning the buzzer on and off. The function could in future studies be used to create different rhythmic patterns based on specific distance thresholds. Including

this function in the project's lines of code ensures that the buzzer's behavior is time-controlled and adjustable depending on the distance measurement.

The time is stored as the variable *duration* with each loop. This variable is then converted into and stored as the distance value using (1). Finally, the calculated distance is printed to the Serial Monitor, which allows the users to see the real-time distance measurement on their computer screen.

Overall, this code is written to successfully use an ultrasonic sensor to measure distance and control an external device (like a buzzer) in response to varying distances, making it a simple and effective example of sensor-based automation in Arduino projects.

B. Assembling of the Breadboard Circuit

One of the most important aspects affecting the success of any breadboard circuit project is the correct placement of the components on the circuit board - and this project is no different. The device uses eight different wires of varying length and color, an ultrasonic sensor with four separate, attached pins (the VCC pin, `trigPin`, `echoPin`, and GND), an electronic buzzer, an Arduino Board, and a wire connector to attach the Arduino to the computer power source.

Directly connected to the breadboard is first the eight wires. The placement is as follows:

Wire #	End 1	End 2
1	Positive Line (Board)	3.3 V (Arduino)
2	Negative Line (Board)	GND (Arduino)

Table 1: Placement of Wires 1 and 2

After the attachment of the first two wires, the ultrasonic sensor was placed onto the breadboard. The first pin of the ultrasonic sensor (the VCC pin and leftmost pin) was placed on line a along the breadboard, with the remaining pins (trigPin, echoPin, and GND), placed directly beside that.

The exact position of these pins, as with all of the other pieces to this circuit, is not immensely important beyond ensuring that they are placed along the correct lines. This is because voltage and current run through the entire column and row that the attachment is connected to. So for example, if a voltage supply is connected via wiring to the positive line on a breadboard, then it will be connected to any other wiring whose end is also connected to the positive line. Similarly, the central part of the breadboard contains vertical columns or rows which are all electrically connected internally, allowing every electronic component connected in a row to share the same voltage [7].

This interconnected grid system allows for a quicker and easier circuit assembly, which is extremely useful for larger-scale projects with greater amounts of wiring and electrical components. Additionally, it allows for flexible circuit building, and for components to be connected throughout the entire breadboard circuit without needing to worry about precise alignment.

Wires three through six were placed onto the breadboard circuit after the sensor was properly positioned.

Wire #	End 1	End 2
--------	-------	-------

3	VCC Sensor Pin (Board)	Positive Line (Board)
4	trigPin on Sensor (Board)	Pin 5 (Arduino)
5	echoPin on Sensor (Board)	Pin 6 (Arduino)
6	GND Sensor Pin (Board)	GND Line (Board)

Table 2: Placement of Wires 3 - 6

Once these pins had been placed, the buzzer was to be positioned strategically along the breadboard. The buzzer, like many electronic components, has a positive end (which in this case and in most, is longer) and a negative end. This is to represent the electrical polarity of the system, which determines the direction of electron flow. Electrons flow from the negative to the positive sides of a component [8].

The positive (longer) end of the buzzer was placed into any row of the f column on the breadboard, while the negative (shorter) end of the buzzer was connected to the corresponding row of the e column on the breadboard. Another way to determine the positive end of this specific buzzer was to look at the top of the piece, where a positive sign (+) was engraved into the plastic corresponding to the positive side. This addition was helpful, but not essential to the component's function and is therefore not always going to be included in every electrical buzzer.

After the buzzer was successfully embedded into the breadboard, the last two

remaining wires were ready to be placed to complete the circuit.

Wire #	End 1	End 2
7	Positive End of the Buzzer (Board)	Pin 4 (Arduino)
8	Negative End of the Buzzer (Board)	GND Line (Board)

Table 3: Placement of Wires 7 and 8

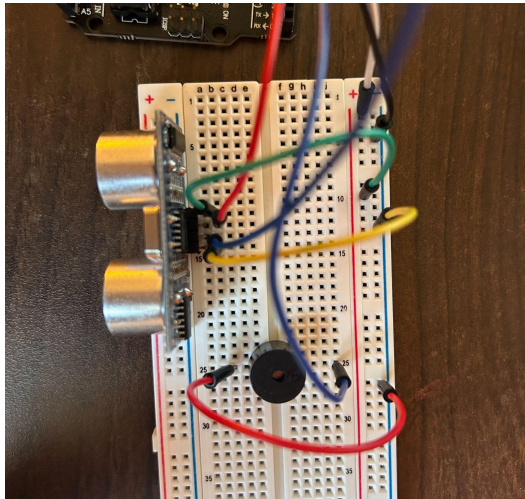


Figure 1: Configuration and Placement of Wires to the Breadboard

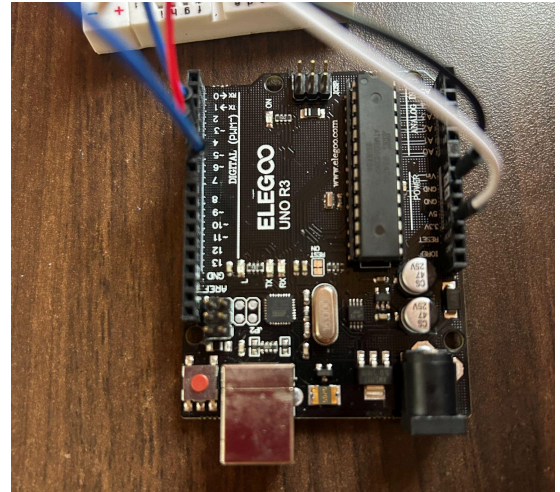


Figure 2: Wire Attachments to the Arduino Board

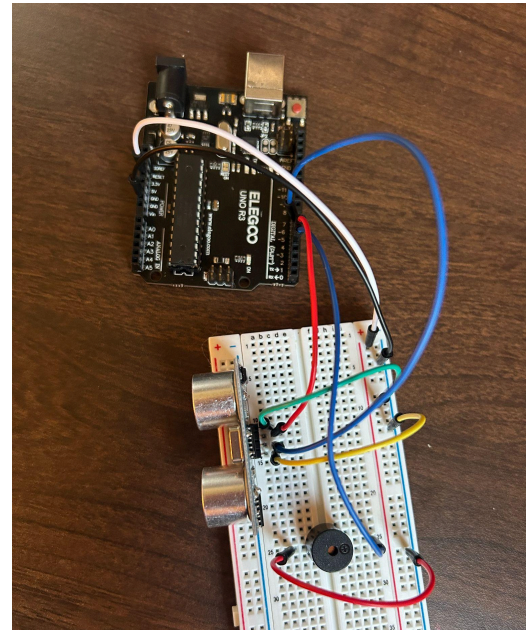


Figure 3: Completed Breadboard Circuit - Overhead View

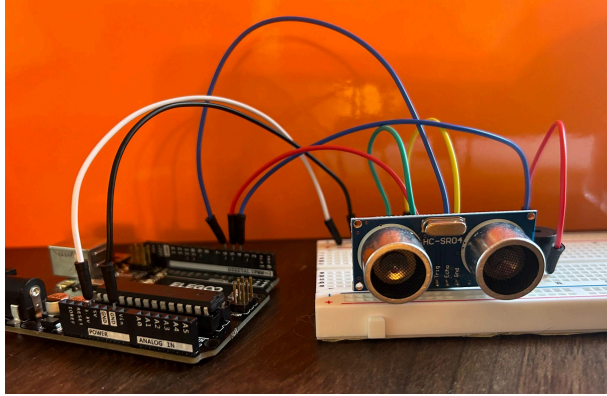


Figure 4: Completed Breadboard Circuit - Lateral View

Figure 1, Figure 2, Figure 3 and Figure 4 all show different perspectives of the completed Arduino and breadboard circuit.

C. Issues Regarding the Circuit Setup

There were a number of different challenges and struggles to overcome when assembling the breadboard circuit. Being that the breadboard used was secondhand (and therefore not exactly in pristine condition), there were a number of different terminal strips throughout the board that did not grab onto the electrical components as strongly or as securely as would have been expected and preferred. However, as this was not the case for every single terminal on the breadboard, the circuit was still able to be completed to success.

Selecting the appropriate wiring was a tedious task, as many of the wires were inadequate lengths, making the connection of longer distances difficult. Some wires were falling apart and not able to be used for this project. The solution to this issue was to select intact wires of appropriate lengths for the given distances.

The setup for this project, though thorough, was slightly messy. The usage of eight separate wires was not immensely difficult

to maneuver. However, when cramped into a smaller section of the breadboard, it created an unorganized look to the circuit and made it somewhat difficult to identify which wires were connected where. The solution to this challenge was to stretch out the placement of the wires. As previously mentioned, the exact position of the wires is not exactly important in a circuit setup - rather, it is necessary to place the components along the correct column, where the voltage and current will be connected to. So, expanding the circuit to create more space and less congestion was not an issue.

Again attributed to the secondhand nature of the Arduino circuit board kit used, there were various components missing. One of these components - the buzzer - was essential to the success of this project and was unfortunately missing in the kit. Therefore, the solution was to borrow a buzzer from another kit.

The original inspiration for this project contained an LED light-based alert system in addition to the electrical buzzer. However, the inspiration used a contraption in which three LED lights (red, yellow, and green) were soldered to a metal piece to mimic a stoplight configuration and then attached to the breadboard. While the kit contained three separate LEDs of the same color, it did not come with a metal piece shaped to fit the lights together exactly nor a soldering kit to attach them. So, both the code and the schematics of the circuit board needed to be altered so as to not include the light alert system.

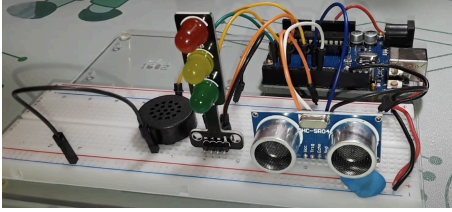


Figure 5: This was the original design that the project aimed to replicate.

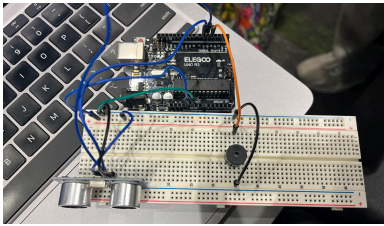


Figure 6: This was the overhead version of the first attempt at the final design.



Figure 7: This is a picture of the testing of the original design and setup.

III. Results and Discussion

The work presented in this project involves the use of an ultrasonic sensor to both measure distances and control an output buzzer device based on those measurements. The main objective of this project was to examine and observe just how varying distances would influence the behavior of the Arduino-controlled device.

The first set of results involved the measurements of distances ranging from close proximity to farther distances. As stated in the methods section, the distances were calculated using the time in seconds it took for a wave pulse to be emitted, hit the nearby object, and return to the sensor. The distance values were calculated by plugging the time duration into (1). Table 4 refers to a trail run of distances

calculated using (1), using times of 100 - 150 μs (or 0.0001 seconds), in increments of 1 μs .

Time (μs)	Distance (cm)
100	1.700
110	1.870
120	2.040
130	2.210
140	2.380
150	2.550

Table 4: Time and Distance Values

As expected, the ultrasonic sensor accurately measured the distance to objects within its sensing range. The buzzer's response was programmed to change based on the distance:

Distance (cm)	Electrical Buzzer Output
Too Close: 0 - 50	Loud, continuous noise
Medium: 51 - 150	Short pulses of noise, delay depends on distance
Good: 151 - 200	Pulses of noise with 1 second delays
Too Far: > 200	No noise

Table 5: Electrical Buzzer Output based on Specific Distances

A. Sources of Error

Issues in the results arose because of the larger magnitudes of distance in the code. Because this project was done on a small scale (in other words, not done with a life-size car and sensor but rather a small Arduino kit and breadboard), the distances were very extreme.

To fix this problem, slight alterations were made to the for loops within the code. Each range of distances was decreased by a factor of 10; Too Close range distances were changed from 0-50 cm to 0-5 cm, Medium range distances became 5.1-15 cm, Good range distances were changed to 15.1-20 cm, and lastly, Too Far distances were changed to anything over 20 cm.

Another challenge that occurred with the inaccuracy of the sensor. The “car” used to mimic an actual car in this project was the projection of a car image on an iPhone. An experimenter was required to hold and move the iPhone manually, which inevitably created an unwanted object within the sensor’s detection. Furthermore, there were a number of different potential sources of object detection by the sensor. The experiment was done in a room, with various moving people, chairs, desks, and more that could have set off the sensor. There may have been unaccounted-for sound waves in the air that reached the sensor before the emitted waves did. There might have been one hefty dust particle floating through the air that could have altered the emitted pulse’s path. All of these might have potentially led to an unwanted detection. To resolve this issue, the sensor’s path was cleared of any and all visible obstacles of reasonable distances.

Overall there was great success in the design and meeting the desired goal. The Arduino code and setup created a proper Assistive Parking Sensor to help keep people safe and avoid damaging their cars.

There were many areas for improvement both on the completion and methodology of bringing the device to fruition and the capabilities of this design. For future work, coming in with a more specific goal and completing more research on what tools were given and what was within the realm of possibilities to complete, would have made this process smoother and faster.

This work is very important as it allows for the safety of cars to increase and help people potentially save hundreds of dollars. Car accidents are such a prevalent problem, and by integrating technology with safety there can hopefully be a big turnaround in these statistics. Car sensors are also a good tool to learn spatial awareness while driving and to keep drivers alert when they may have spaced out and could run into something.

This also opens the door for future improvements to be made to build off of this simpler idea. By understanding and solidifying the basics, one can then go on to find ways to continue this progress in new ways. Some of these future ideas include being able to distinguish types of objects - stationary and moving, and the ability of assistive parking to create a map for users to have or provide driving suggestions, and it can potentially be applied to things beyond cars: Forklifts, or temporary attachable features for moving furniture.

Overall this project sets up many possibilities in combating many of the safety challenges that come with driving to keep

IV. Conclusions

people safe, save them money, and has the potential to expand beyond the household.

V. Acknowledgments

The original inspiration for the proper design setup: T. M. Iqbal, "What is Machine Learning? Machine Learning for Beginners," YouTube. [Online]. Available: https://www.youtube.com/watch?v=t-UTq2N0e_A. [Accessed: Dec. 5, 2024].

Professor Dr. David J. Icove and our TA for helping find the proper resources needed.

VI. References

- [1] Nelson MacNeil, "How common are parking lot car crashes?", Nelson MacNeil, 2023. [Online]. Available: <https://nelsonmacneil.com/blog/how-common-are-parking-lot-car-crashes/>. [Accessed: Dec. 5, 2024].
- [2] Stormwise Hail Repair, "Car body damage repair cost," Stormwise Hail Repair, 2023. [Online]. Available: [https://stormwisehailrepair.com/car-body-damage-repair-cost/#:~:text=Minor%20scuffs%20or%20scratches%20%2DJust,Typically%2C%20it's%20\\$100%20to%20\\$300](https://stormwisehailrepair.com/car-body-damage-repair-cost/#:~:text=Minor%20scuffs%20or%20scratches%20%2DJust,Typically%2C%20it's%20$100%20to%20$300). [Accessed: Dec. 5, 2024].
- [3] Bankrate, "Will an accident increase my insurance premium?," Bankrate, 2023. [Online]. Available: <https://www.bankrate.com/insurance/car/rates-after-an-accident/#will-an-accident-increase-my-insurance-premium>. [Accessed: Dec. 5, 2024].
- [4] MaxBotix, "How ultrasonic sensors work," *MaxBotix*, 2023. [Online]. Available: <https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work?srltid=AfmBOorjxpngLDP>

[5bMaE-T5iQrSCtA4rMqkf74LGa5bvgINnz1jOXRMM](https://www.maxbotix.com/blogs/blog/how-ultrasonic-sensors-work?srltid=AfmBOorjxpngLDP). [Accessed: Dec. 5, 2024].

[5] Arduino, "setup()," Arduino, 2023.

[Online]. Available:

<https://www.arduino.cc/reference/tr/language/structure/sketch/setup/>. [Accessed: Dec. 5, 2024].

[6] CircuitBasics, "Using loops in Arduino programming," CircuitBasics, 2023. [Online]. Available:

<https://www.circuitbasics.com/using-loops-in-arduino-programming/#:~:text=Loops%20are%20used%20to%20control,enters%20the%20loop%20or%20not>. [Accessed: Dec. 5, 2024].

[7] Science Buddies, "How to use a breadboard," Science Buddies, 2023. [Online]. Available:

<https://www.sciencebuddies.org/science-fair-projects/references/how-to-use-a-breadboard>. [Accessed: Dec. 5, 2024].

[8] Squishy Circuits, "What is polarity?," Squishy Circuits, 2023. [Online]. Available: <https://squishycircuits.com/pages/what-is-polarity/#:~:text=Have%20you%20ever%20looked%20at,side%20to%20the%20positive%20side>. [Accessed: Dec. 5, 2024].

ORCHID:

Sophia Hickey's QR Code



Ava Delgra's QR Code

