Sound based path planning for Next-Gen Smart Vehicular Navigation

A PROJECT REPORT(MIDSEM)

Submitted in partial fulfillment of the requirements for the award of the degrees

of
BACHELOR OF TECHNOLOGY
in

ELECTRICAL ENGINEERING

Submitted by:
LAKSHAY (210002050)
SHASHANK SHARMA (210002069)

Guided by:
DR. SUMIT GAUTAM

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE

INTRODUCTION:

With the advancement in technology of **autonomous vehicles** and smart transportation, new methods of navigation are very much needed. In this project, we will be focusing on sound-based path Planning where the sound signals from the buzzers will be utilised for decision making and path planning of our autonomous vehicle.

By using **sound sensors (KY-038)**, **Arduino**, **buzzers** and **advanced signal processing**, our vehicle will be able to run on the specific desired path we want it to run on. Such kinds of methods are very usefull in scenarios where traditional methods or GPS-based navigation may be hindered, such as bad weather, low-visibility conditions, parking lots, or environments with limited connectivity.

This project explores the integration of sound-based navigation into the broader framework of intelligent transportation systems, highlighting its potential to enhance safety, efficiency, and adaptability in **autonomous vehicles** and **smart city infrastructure**.

MOTIVATION:

The rapid advancement in autonomous vehicles and intelligent transportation systems calls for innovative methods to navigation and decision-making. GPS, cameras and Lidar are proven technologies but work only under certain conditions. The primary goal of this project is to work towards abolishing all the existing problems in order to make vehicular navigation more versatile and adaptable.

1. **Difficult Environments:** Audio localization offers a reliable alternative to sound when visual and lidar sensors may struggle.
2. **Safety and Communication:** Sound-based navigation is modelled on this feature of the real biosphere and enables behaviorial response of vehicles to sound sensed through microphones. It also improves safety on the road — as it allows vehicles to respond to auditory prompts and notifications from their surroundings, which helps avoid accidents and communicate launches with other exchangers and pedestrians
3. **Cost Efficiency:** this system is much cheaper in comparison to high-end **visual systems** like cameras and lidar. This makes sound-based navigation much affordable
4. **Enhancing Autonomous Decision-Making:** This enhances environmental awareness (outside of GPS and LIDAR) by allowing vehicles to adapt their behaviour and path in real time based on auditory information increasing decision-making efficiency thus reducing the dependency on systems that require complex data pipeline for noise robust regression.
5. **Integration with Smart City Infrastructure:** This is the future of smart cities that will undoubtedly include vehicle to infrastructure operations. Sensors located in intersections then transmit this information to the intersection control system, which activates audio signals that inform visually impaired drivers of when and where they can safely enter traffic.

THEME OF WORK:

This project is basically based on the agile **navigation** of autonomous vehicles with the help of sound signals. It explores the possibility of "whether sound can be a robust environmental cue to aid or even replace other forms of navigation, such as GPS, cameras and lidar in difficult conditions."

This project looks into the research of planning a **sound-based** path by which it focuses on how to design, realize and optimize a system able to make vehicles able traverse than moves according to audio hints example beeps as in our instance. This makes it easier for a vehicle to deal with dynamic environments and unpredictable traffic situations.

OBJECTIVE:

The aim of this project is to create an **expressive vehicular navigation** system which enables to autonomously move our vehicle other than the GPS navigations and older technologies. Thus, increasing the use case, efficiency of the autonomous navigation domains. Sound sensors, plus appropriate **signal processing algorithms** to enable the vehicle to react to transient environmental cues e.g., beeps in our case, looked into as an alternative approach for assisting the traditional visual or GPS-based navigation systems.

LITERATURE SURVEY:

1) **Sound Sensors**: there are a lot of sensors available. So a lot of research had to be done on what sensor would be best in our case [https://www.watelectronics.com/sound-sensor/].
2) **Motor Drivers**: Similarly, there are a lot of motor drivers as well. So, choosing them wisely and learn how they are used is very important [https://robocraze.com/blogs/post/what-is-motor-driver].
3) **Use case**: Basically whatever we are creating must be because of the applications it has across various fields [https://www.startus-insights.com/innovators-guide/autonomous-vehicles-startups/].
4) **Challenges Involved**: When we try to apply such concepts in real world, there are a lot of challenges that we would be facing [Navigating the Complexities: Top 10 Challenges on the Road to Autonomous Vehicle Adoption by Vasantharaj G]

WEEK 1: Buzzer Beeping Based on Button Press

Introduction

This project demonstrates how to control a buzzer using four push buttons, where each button triggers a different number of beeps. The primary objective is to simulate an interactive system where button presses result in buzzer beeps.

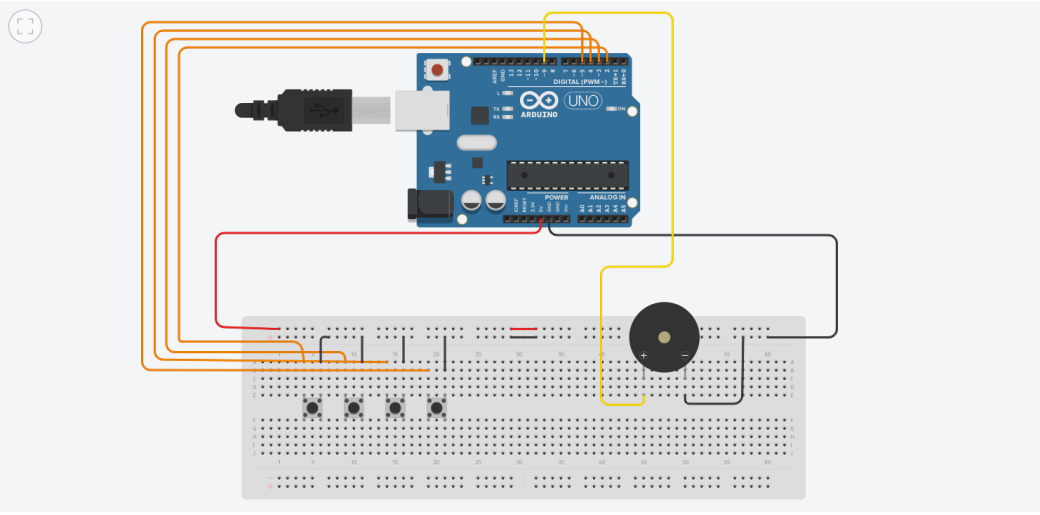
Components Used

1. **Arduino Uno** – The microcontroller board used to execute the code and control the buzzer and buttons.
2. **Passive Buzzer** – The output device used to produce beeping sounds.
3. **4 Push Buttons** – The input devices used to trigger the buzzer beeps.
4. **Breadboard** – Used for building the circuit without soldering.
5. **Jumper Wires** – To make connections between the components and the Arduino board.

Concepts Involved

1. **Arduino Basics:** The Arduino Uno is a microcontroller that runs simple to complex programs, interacting with sensors and actuators. The buttons act as digital inputs, and the buzzer is controlled via a digital output pin.
2. **Push Buttons:** Instead of pull-down resistors, you can enable the internal pull-up resistors in the Arduino code, which simplifies the wiring. You would also connect the other side of the button to GND, and pressing the button will give a LOW signal.
3. **Buzzer Basics:** A passive buzzer generates a sound when voltage is applied. It has two pins: positive (anode) and negative (cathode). Controlling the buzzer involves turning it on and off for specified durations to create beeping sounds.
4. **Serial Communication:** Serial Monitor Debugging is used to print messages for each button press, aiding in debugging and monitoring real-time input.

Circuit Diagram



Arduino Code

```
// Define button pins
const int button1Pin = 2;
const int button2Pin = 3;
const int button3Pin = 4;
const int button4Pin = 5;
// Define Buzzer pin
const int buzzerPin = 9;
// Beep duration in milliseconds
const int beepDuration = 200;
const int buzzerFrequency = 1000;  // Frequency in Hz (1000 Hz = 1 kHz tone)
void setup() {
  // Initialize button pins as inputs with internal pull-up resistors
  pinMode(button1Pin, INPUT_PULLUP);
  pinMode(button2Pin, INPUT_PULLUP);
  pinMode(button3Pin, INPUT_PULLUP);
  pinMode(button4Pin, INPUT_PULLUP);
  // Initialize serial communication for debugging
  Serial.begin(9600);
}
void loop() {
  // Check button states and perform beeps when LOW (button pressed)
  if (digitalRead(button1Pin) == LOW) {
    Serial.println("Button 1 pressed");
    beep(1);
  } else if (digitalRead(button2Pin) == LOW) {
    Serial.println("Button 2 pressed");
    beep(2);
  } else if (digitalRead(button3Pin) == LOW) {
    Serial.println("Button 3 pressed");
    beep(3);
  } else if (digitalRead(button4Pin) == LOW) {
    Serial.println("Button 4 pressed");
    beep(4);
  }
}
// Function to beep the passive Buzzer n times
void beep(int times) {
  for (int i = 0; i < times; i++) {
    tone(buzzerPin, buzzerFrequency);  // Start sound at the specified frequency
    delay(beepDuration);
    noTone(buzzerPin);  // Stop the sound
    delay(beepDuration);
  }
}
```

Working Principle

The project works based on the interaction between the Arduino and the buttons:

- Each button is associated with a unique number of beeps (1, 2, 3, or 4).
- When a button is pressed, the Arduino detects a HIGH signal on the respective input pin, triggering the corresponding number of buzzer beeps.
- Each beep consists of turning the buzzer on for a specified duration (beep Duration), followed by turning it off for the same duration.
- The project provides a simple interface for auditory feedback via the buzzer, making it easy to track how many times a button is pressed.

Applications

This project serves as an introduction to Arduino's digital I/O control and can be expanded for various applications, such as:

- **User Input Interfaces:** Implementing different responses based on user input, such as interactive sound or alarm systems.
- **Control Systems:** Using buttons to control audible feedback or triggering actions based on the number of beeps.

- **Game Design:** Implementing simple game mechanics where pressing buttons gives auditory feedback via a buzzer.

WEEK 2: SOUND SENSING AND COUNTING USING KY-038 SENSOR

Introduction

This section demonstrates how the KY-038 sensor detects the beeps made by buzzer. The goal is to detect beeps over a specified time interval (5 seconds), count them, and output the total number of beeps within that time period on the serial monitor. Later on, we will also discuss few applications of this section.

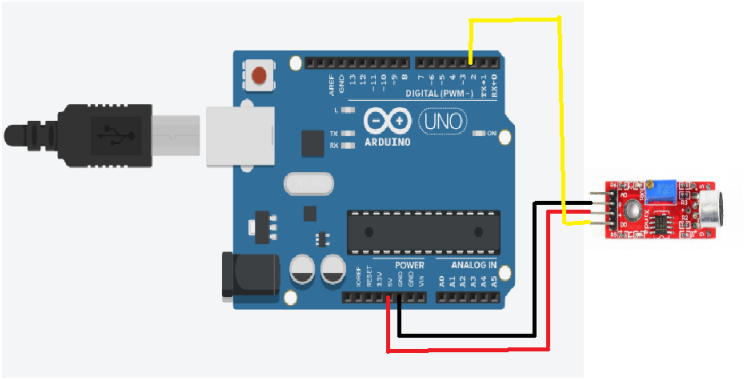
Components Used

1. **Arduino Uno** – The microcontroller board used to interface with the sound sensor.
2. **KY-038 Sound Sensor** – A sound detection module that senses sound intensity and outputs a digital signal when sound exceeds a threshold.
3. **Jumper Wires** – Used for making connections between the sound sensor and the Arduino.
4. **Breadboard** – For easy wiring and connections.

Concepts Involved

1. **Arduino Basics:** The KY-038 sound sensor outputs a digital signal to indicate when the sound exceeds the preset threshold. Where the built-in LED on pin 13 is used as a visual indicator for sound detection which basically acts as the digital output.
2. **Sound Detection:** KY-038 Sound sensor detects sound based on intensity and outputs a HIGH signal when the sound surpasses the threshold. The threshold can be adjusted.
3. **Time-based Counting:** millis (), function is used to keep track of elapsed time without blocking other code execution.
4. **Debouncing:** A short delay of 500 milliseconds is added after detecting a beep to avoid counting multiple times for the same beep.

Circuit Diagram:



Arduino Code:

```
const int soundSensorPin = 2; // KY-038 digital output connected to digital pin 2
const int ledPin = 13;        // Built-in LED for indication
int beepCount = 0;            // Variable to store the count of beeps
unsigned long startTime = 0;  // Variable to store the start time of counting
const unsigned long interval = 5000; // Time interval of 5000 milliseconds (5 seconds)
void setup() {
  pinMode(soundSensorPin, INPUT);  // Set the sound sensor pin as input
  pinMode(ledPin, OUTPUT);         // Set the LED pin as output
  Serial.begin(9600);              // Start serial communication for debugging
  startTime = millis();            // Initialize the start time
}

void loop() {
  int sensorValue = digitalRead(soundSensorPin); // Read the digital output from KY-038
  if (sensorValue == HIGH) {   // Check if the sound intensity exceeds the threshold
    beepCount++;               // Increment the beep count
    Serial.println("Beep detected!");  // Output to serial monitor for debugging
    digitalWrite(ledPin, HIGH);        // Turn on LED to indicate beep detection
    delay(500);                        // Debounce delay to prevent multiple counts for the same beep
  } else {
    digitalWrite(ledPin, LOW);         // Turn off LED if no sound is detected
  }
  if (millis() - startTime >= interval) {
    Serial.print("Total beeps detected in the last 5 seconds: ");
    Serial.println(beepCount); // Output the total count of beeps
    beepCount = 0;             // Reset beep count for the next interval
    startTime = millis();      // Reset the start time for the next interval
  }
  delay(30);  // Small delay to avoid overly frequent checking
}
```

Working Principle

- **Sound Detection:**The KY-038 sound sensor monitors ambient sound intensity. When the sound exceeds the preset threshold (which can be adjusted with the potentiometer), it sends a HIGH signal to pin 2 of the Arduino.
- **Beep Counting:** The Arduino counts each beep detected by incrementing the beepCount variable whenever the sensor output goes HIGH.
- **Time Interval Measurement:** The program uses the millis() function to track a 5-second time interval. After each interval, the total number of beeps is printed to the serial monitor, and the counter is reset for the next interval.
- **LED Indicator:** The onboard LED (connected to pin 13) lights up whenever a beep is detected, giving a visual indication of sound detection.

Applications

Few of the important applications of this section are:

- **Noise Monitoring Systems:** Detecting and counting noise events over time.
- **Security Systems:** Monitoring beeps or sounds in a specific environment for security purposes.
- **Interactive Sound-based Projects:** Where sound triggers specific actions or events.

WEEK3: 3D Printed Parts for a Vehicle – Chassis and Tires And Assembling Of Vehicle

Introduction

This section focusses on creating the chassis, tires and other parts of our vehicle using 3D printing technique. We will also discuss about the assembling of various parts on the chassis in this section. And at last, we will also get to know few advantages and applications of 3D printing.

Components Used

1. **3D Printer**– A 3D printer was used to print the tires.
2. **PLA Filament (Polylactic Acid)** – A commonly used biodegradable filament for 3D printing.
3. **ply** – The main structural component of the vehicle, providing support for components.
4. **DC Motors or Servo Motors** – Used to drive the wheels.
5. **Screws and Nuts** – For assembling sensors, motors and other parts.

Designing the 3D Models

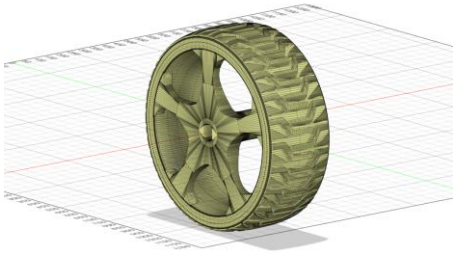
1. **Chassis Design:** Using fusion 360, the chassis was designed, ensuring it had the proper dimensions to fit the motors, batteries, and other electronics.
2. **Tire Design:** The tires are designed with special thread patterns on its cylindrical section which would help in providing more grip while in use. Also, the size of tires are based on the chassis design and the motor we used. The hub i.e. the inner part of the tire was designed to fit the motor shafts securely, ensuring that the tires do not slip while rotating.

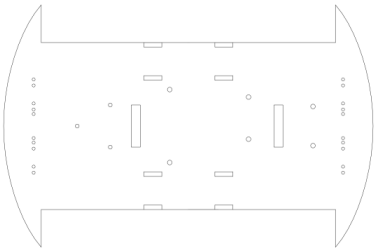
3D Printing Process

1. **Printing:** chassis is made by using cutting technique of a ply by passing the design made in fusion 360. Whereas the tires were 3D Printed. Each tire took around a few hours, depending on the size and tread pattern.
2. **Post-Processing:** After printing, the chassis and tires were sanded lightly to remove rough edges and improve the fit of the components.

The Designs made in FUSION 360:

Tires and chassis:





Advantages of 3D Printing for Vehicle Parts

1. **Customization**: we can create specific design based on our needs and can resize or change wherever we want.
2. **Rapid Prototyping**: This allows us to test much faster.
3. **Cost-Effectiveness**: Very cheap compared to other traditional ways.
4. **Lightweight Construction**: It is one of the best techniques where weight is a concern.

Applications

In this section, some techniques for making different types of vehicle prototypes have been discussed as outlined below.

1. **Arduino-controlled Robots**: The chassis and tires can be implemented to develop an autonomous or remote-controlled vehicle with running on Arduino and motor drivers.
2. **Custom RC Cars**: Hobbyists can design and print custom vehicles using specific tire designs and chassis layouts specifically for a remote-controlled car.
3. **Educational Robotics**: Students and makers can be taught vehicle dynamics, 3D design, and manufacturing by using 3D printed parts in a classroom or lab.

WEEK 4: SOUND CALIBRATION AND FILTERATION OF BUZZER BEEPS FOR SOUND SENSING

Introduction

Well in this project are going to have multiple buzzers, so we need to make sure that the sensor detects only that beep sound that we want it to detect. We need to make sure that it does not detect any external noise as well. In this particular project our aim is to make sure that the above-mentioned task is done.

Concept involved

Well one thing that we need to know is that the KY-038 sensor detects two types of signals, which are the Analog and the Digital signal. Actually, we can decide what output signal we require in our project.
Different techniques that may be applied for sound filtration are:

1. Threshold Adjustment: The KY-038 sensor has a pot (kind of screw) on it. By rotating that pot we can adjust the sensitivity of the sensor but we can set any threshold on it just by rotating it. More will be the sensitivity, more will be its chances of even detecting a very small sound.

2. Frequency Filtering (Hardware): As we have studied there are band pass filters which just passes only a particular band of frequencies through it. Thus, this helps us in frequency filter by not adjusting the code in Arduino.

3. Software Filtering: for detecting some specific frequencies we have to implement a Fast Fourier Transform (FFT) algorithm in your Arduino code to analyse the frequency components of the sound detected by the KY-038 sensor.

4. Time-Based Filtering: Basically in this we can use the intervals in between two beeps as a medium of filtering. If the interval would be lesser than a particular set timer then we will just ignore that beep

5. Noise Filtering: Background noise would be reduced if you run an average on several readings or apply a moving average filter to your code that would actually smooth out the signal and avoid detection of false positives.

Since it's already discussed that we use digital output from the sensor, therefore by adjusting the sensitivity of the KY-038 sensor by using the potentiometer we will filter most of the other digital signals.
For the Analog signal we would require to set threshold and use the FFT algorithm in the code shown below:

```
const int soundSensorPin = A0;
const int threshold = 500;
const int minFrequency = 900;
const int maxFrequency = 1100;
void setup() {
  Serial.begin(9600);
  pinMode(soundSensorPin, INPUT);
}
void loop() {
  int soundLevel = analogRead(soundSensorPin);
  if (soundLevel > threshold) {
    if (detectBeepFrequency(minFrequency, maxFrequency)) {
      Serial.println("Beep detected!");
    }
  }
  delay(100);
}
bool detectBeepFrequency(int minFreq, int maxFreq) {
  return true;
}
```

Applications

1. Sound-Based Security Systems

2. Medical and Health Monitoring

3. Gesture Control Using Sound

4. Wildlife Monitoring

5. Industrial Noise Monitoring