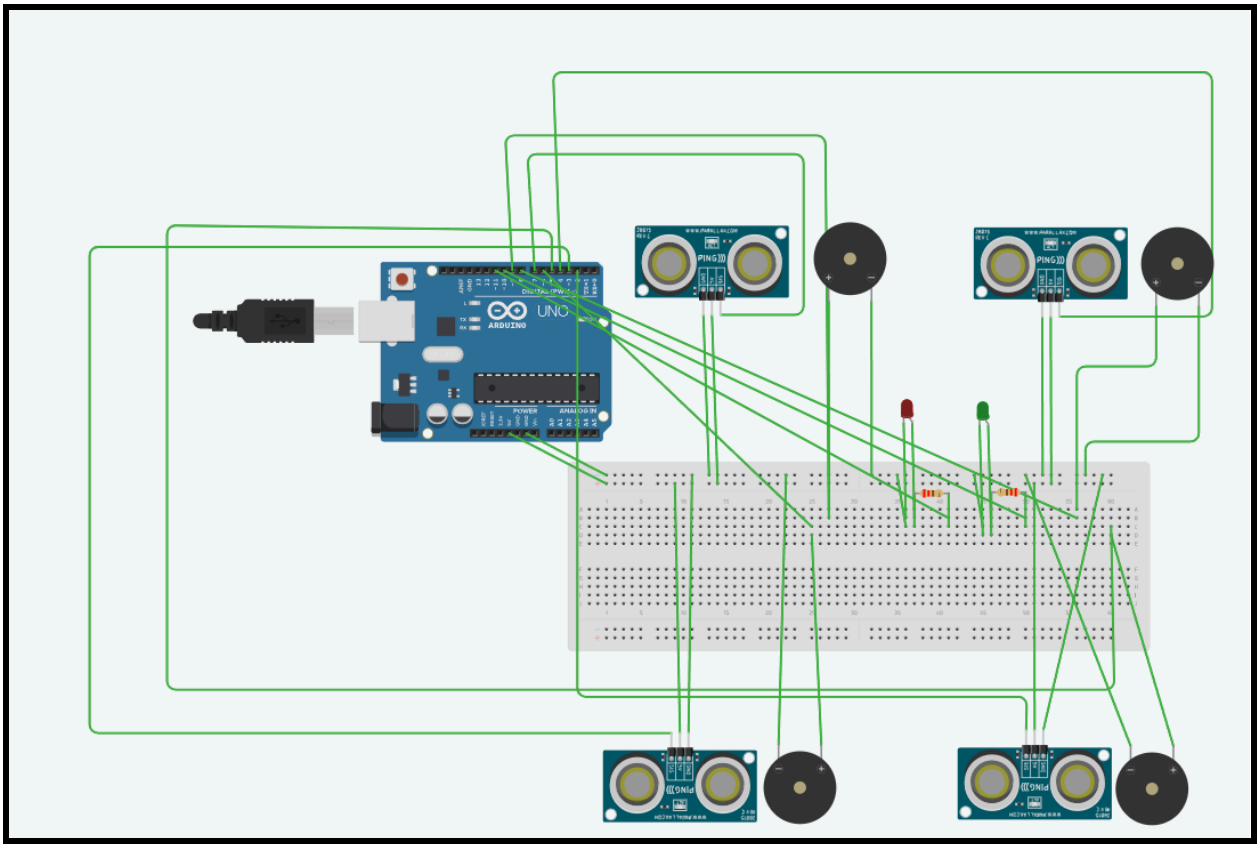


Driving Assistance System

Flori Kusari

Rron Dermaku

Intro To Robotics and Intelligent Systems

Lab Report Project

Overview

The Driving Assistance System is a project to make parking and driving cars easier. The system is user-friendly and targeted to increase safety and efficiency. The system utilizes ultrasonic sensors and speakers to achieve its goal by providing real-time feedback to the user on all approaching objects. The feedback is received through the speakers for each respective ultrasonic sensor and through visual representations displayed to the user using Python-based graphic representations of the distance of objects from each sensor. The data from these readings can also be used to predict the time during which parts will need to be replaced based on how often they are used.

Characteristics

1. User-Friendly
2. Automatic
3. Safe
4. Efficient
5. Self-Diagnostic

How it works

The core of the Smart Parking Assistant System lies in its ultrasonic sensors, positioned strategically to cover all sides of the vehicle. These sensors measure the distance between the car and nearby objects and display it to the user. Once the distance becomes lower than the allowed limit, a sound is let out by the speaker corresponding to the sensor which informs the driver to be careful. If the distance is lower than 1 meter, then a green light will turn on while the speaker rings informing the driver that the object is close but it is still relatively safe, however, once the distance becomes less than 20 cm a red light is turned on indicating that it is incredibly dangerous and that the driver should be cautious.

Using the system

Watch the computer screen's graphical user interface and note how the piezo speakers' beeping frequency varies as you maneuver your car into a parking spot. Make precise measurements of the distance between your car and nearby obstructions by using these clues provided to you by the system. The technology is designed to be user-friendly and it will alert you when you are approaching an object too closely with a rapid beeping sound

and it will tell you the rough distance through visualization and the alert lights (Green for distances less than 1 meter and Red for distances less than 20cm).

Components

Name	Quantity	Official Component Name
UArduino-Main	1	Arduino Uno R3
PINGLeft-Back-Sensor PINGRight-Back-Sensor PINGLeft-Front-Sensor PINGRight-Front-Sensor	4	Ultrasonic Distance Sensor
PIEZOLeft-Back-Speaker PIEZOLeft-Front-Speaker PIEZORight-Front-Speaker PIEZORight-Back-Speaker	4	Piezo Speaker
DAlertLED-Red DAlertLED-Green	2	LED
RGreen-LED-Resistor, RRed-LED-Resistor	2	220 Ω Resistor

For a visual of all components see the picture on the first page

Component Usage

Arduino Uno R3:

Function: The brain component of the project. It controls the sensors and speakers, reads sensor data, activates the speakers and LEDs, runs the code written for the project, and implements it through the components connected with it.

Connection: Connects to all components of the project.

Ultrasonic Distance Sensors:

Function: The sensor works by sending a series of sound waves and recording the time it takes for them to return after bouncing off an object. The controller then outputs this time interval encoded as the width of a voltage pulse (that can be read by Arduino and interpreted as distance).

Connection: Each sensor's VCC and GND pins are connected to the Arduino's 5V and the ground (GND). The signal (SIG) pin is connected to a specific digital pin on the Arduino for data reading.

Piezo Speakers:

Function: As the name suggests the speakers produce sound signals. Depending on the distance measured by the ultrasonic sensors, the piezo speakers produce different frequencies of beeps.

Connection: For signal control, one lead is linked to a digital pin on the Arduino, and the other lead is connected to the ground (GND).

LEDs:

Function: Provide visual alerts. The red LED might indicate danger, and the green LED indicates a safe distance.

Connection: The anodes of the LEDs are connected through a 220 Ω resistor to a digital pin on the Arduino. The cathodes are connected to the ground (GND).

220 Ω Resistors:

Function: The resistors limit the current to the LEDs.

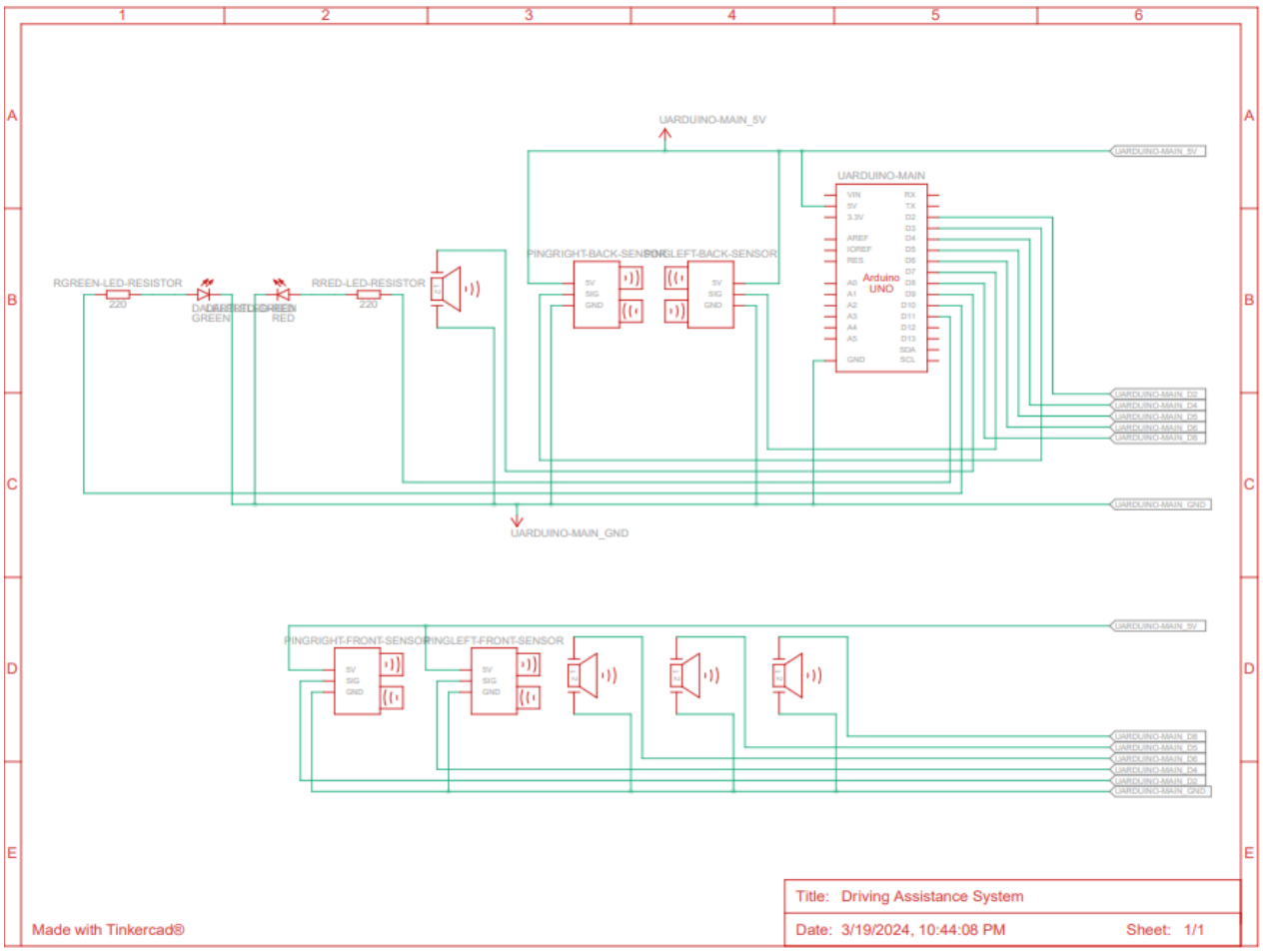
Connection: Each resistor is connected to the anode of an LED and on the other end to the digital control pin on the Arduino.

Breadboard:

Function: The breadboard is a construction base used to build prototypes of electronic circuits. The breadboard assists the circuit constructor in making connections between components through internally connected inner columns and outer rows.

Connection: In the Driving Assistance System, the breadboard facilitates the connections between the Arduino Uno R3 board and the other components (ultrasonic sensors, piezo speakers, LEDs, and resistors).

The Circuit Diagram



The Diagram of the Project's Circuit

The diagram showcases the project as a circuit. All components are connected to the Arduino Uno R3 board, including the breadboard used to assist in efficiently connecting all components. The diagram makes it easy to follow these connections and understand all relations between components.

Arduino Code

The project needs instructions to understand how to behave and these instructions must be implemented through code. The Arduino Uno R3 board functions as the compiler and the implementor of this code. The code is written in C/C++ code and the Arduino Uno R3

board receives the code through a USB 2.0 cable and then implemented through the board. The code shown below will run until forced to stop when the simulation ends.

```
// Define pins
const int sensorPins[] = {2, 3, 4, 7}; // Sensor signal pins
const int speakerPins[] = {5, 6, 8, 9}; // Speaker pins

// Define pins for the LEDs
const int ledPinGreen = 10;
const int ledPinRed = 11;

void setup() {
  // Initialize sensor signal pins as OUTPUTs using for loop
  for(int i = 0; i < 4; i++) {
    pinMode(sensorPins[i], OUTPUT);
  }
  // Initialize speaker pins as OUTPUTs using for loop
  for(int i = 0; i < 4; i++) {
    pinMode(speakerPins[i], OUTPUT);
  }
  // Initialize LED pins as OUTPUTs
  pinMode(ledPinGreen, OUTPUT);
  pinMode(ledPinRed, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  // Array for distances measured
  long distances[4];

  for(int i = 0; i < 4; i++) {
    distances[i] = measureDistance(sensorPins[i]);
    // Control speaker for corresponding sensor
    if(distances[i] < 20) {
      // If object is 20cm away, intensify beeps
      tone(speakerPins[i], 1000);
    } else if(distances[i] < 100) {
      // If object is 21-100cm, output calm beeps
      tone(speakerPins[i], 1000, 200);
    } else {
      // If object is further, no need for beep.
      noTone(speakerPins[i]);
    }
  }

  // Find the minimum distance
  long minDistance = distances[0];
  for(int i = 1; i < 4; i++) {
    if(distances[i] < minDistance) {
      minDistance = distances[i];
    }
  }

  // The Red LED triggers for <20cm and Green LED for <100cm
  if(minDistance < 20) {
    digitalWrite(ledPinRed, HIGH);
    digitalWrite(ledPinGreen, LOW);
  } else if(minDistance < 100) {
    digitalWrite(ledPinRed, LOW);
    digitalWrite(ledPinGreen, HIGH);
  } else {
    digitalWrite(ledPinRed, LOW);
    digitalWrite(ledPinGreen, LOW);
  }

  // Send distances data to the serial port
  for(int i = 0; i < 4; i++) {
    Serial.print(distances[i]);
    if(i < 3) {
      // Separate the distances with commas
      Serial.print(",");
    }
  }
  // End the line for Python reader
  Serial.println();

  // Delay between measurements
  delay(50);
}

// Function to measure distance using an ultrasonic sensor
long measureDistance(int pin) {
  // Set the trigger pin to LOW
  digitalWrite(pin, LOW);
  // Wait for 2 microseconds
  delayMicroseconds(2);

  // Setting the pin HIGH for 10 microseconds
  digitalWrite(pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(pin, LOW);

  // Switches the pin to INPUT mode to listen for the echo
  pinMode(pin, INPUT);

  // Measure the time it takes for the echo to return
  long duration = pulseIn(pin, HIGH);

  // Switches the pin back to OUTPUT mode
  pinMode(pin, OUTPUT);
  // Calculate distance in cm using formula
  long distance = duration * 0.0343 / 2;

  // Return the calculated distance
  return distance;
}
```

The Arduino Code

The code is made up of two main functions which are the *setup* and the *loop* void functions. The setup function initialized the pins of the components as output for them to be used later in the code. The loop function is more complex and it contains instructions on most of the operations carried out in the project. The sensors measure the distances through a function that uses a formula to turn the readings into cm. The distances are stored in variables which are then used for other functions and put through conditional statements that decide which LEDs or speakers, if any, should be triggered. The stored values are then output through the serial port in a specific format which is read by the Python Code which is responsible for the visual representation of the readings that are displayed to the user in real-time.

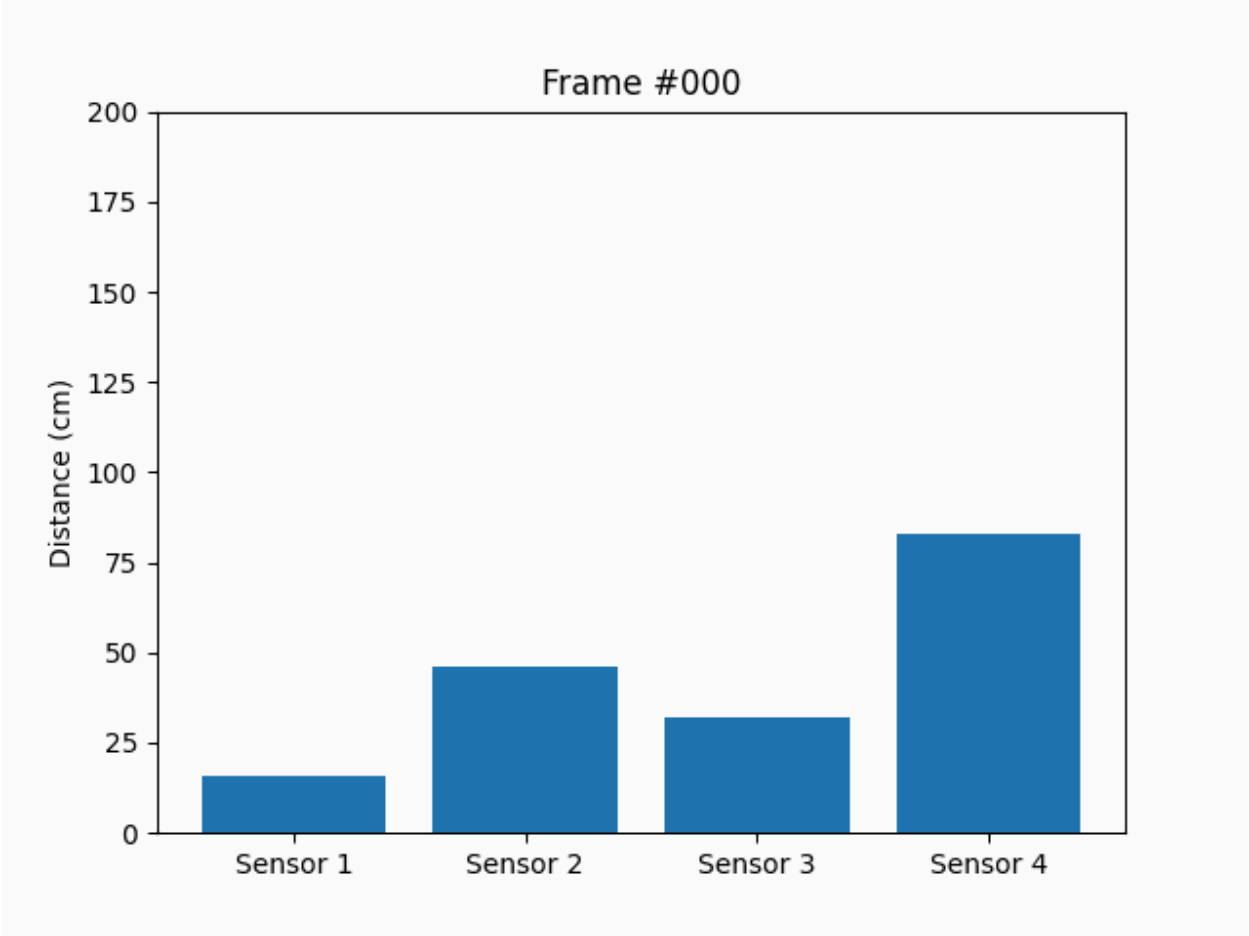
Python Code

```python
import serial
import matplotlib.pyplot as plt

# Set up serial connection
ser = serial.Serial('COM3', 9600)

def read_serial():
    if ser.in_waiting > 0:
        # Reads a line of data
        data = ser.readline().decode('utf-8').rstrip()
        return data
    return None

def plotmaker(distances):
    # Setting up the graph
    fig, ax = plt.subplots()
    ax.bar(['Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4'], distances)
    ax.set_ylim(0, 200)
    ax.set_ylabel('Distance(cm)')
    ax.set_title('Readings from Ultrasonic Sensors')
    plt.show()

# Read data from the serial port
serial_data = read_serial()

if serial_data:
    # Parse the serial data
    # Splits the 4 measurements using
    readings = serial_data.split(',')
    distances = [float(dist) for dist in readings]  # String -> Float
    # Generate the plot
    plotmaker(distances)
else:
    print("****No data received. Check the serial connection.****")

# Close the serial connection
ser.close()
```

The Python Code

The Python code connects the serial port so that it can read the output of the sensors in real time. The code works by reading the data for each sensor and storing it then creating a bar plot to show the distance for all four simultaneously. The visual below shows only one reading, but the code below will enable the readings to be shown at all times if implemented. The code checks if there is data to be read and if the data is present, then it separates it into 4 readings which are used for the visual representation. After the data is separated the graph is output for the user and the serial connection is closed.

Visual Representation



Graph Displayed By Python Code

The graph above presents a visual representation of the distances measured by the ultrasonic sensors and processed by the Python Code. The graph would update with each measurement done by the ultrasonic sensors and displayed to the user. This visual representation combined with the LEDs and the speakers is the core of what makes the Driving Assistance System user-friendly.

Conclusion

The Driving Assistance System project was designed and implemented to increase safety in driving vehicles. A system to aid in informed and safe driving by guiding the driver into the system involving Python and Arduino (C/C++) code, incorporated using hardware components, among them Arduino Uno R3 Board, Ultrasonic Sensors, Piezo Speakers, and LEDs.

Going forward, the driving assistance system opens room for innovation through predictive maintenance alerts for speakers based on their use patterns and frequencies, integration of smartphones with the car for improved levels of user interfacing, and adoption of machine learning for adaptive assistance customized for each user's driving style. In essence, the Driving Assistance System realizes not only its initial aims but also opens the way for further elaborations for solving long-lasting driving problems.

Contribution of Members

Flori Kusari: Arduino Code, Arduino Circuit.
Rron Dermaku: Python Code, Written Lab Report.