CS 362 Group Project

Group Members:

- Keegan Bain, kbain4, kbain4@uic.edu
- Shaan Shekhar, sshkeh7, sshkeh7@uic.edu
- Aditya Dev Reddy, aguda4, aguda4@uic.edu

Name of the Project: Car Collison Detection System

Abstract: The Car Collison Detection System will utilize a variety of input (sensors) and output devices (display, phone notification, buzzer, LEDs, etc.) to estimate a possible collision, displaying on the 16x2 display, LEDs, and sending data to the user. And in case of collision, sending notification to the user, and ringing a buzzer. Furthermore, the data can be utilized to tell/predict user driving habits, and suggestions that can help improve driving habits.

Detailed Project Ideas:

1. **Description:** The detection system utilizes an ultrasonic sensor to estimate distance of the object in its radar. The data is them processed to light up an LED depicting that specific range in which the object lies in. Additionally, the LED display displays if the object is very close to the sensor, or at a faraway distance.

   In case the distance is too close to the sensor (i.e. the collision occurs), the buzzer in the system starts, to notify the occurrence of a collision. Moreover, the user's phone, possibly connected via Bluetooth, will have a notification sent about the happening of the collision.

Additionally, all the data collected in real-time will not be thrown away. Instead, we have planned to process all the data available, to predict user's driving habits (i.e. aggressive driving or keeps a safe distance at all times), and suggest changes that can be implemented to ensure good driving.

2. **Project Design:**

   We plan on taking the input Ultrasonic sensor and using the data from the sensor to perform a variety of tasks including lighting up a specific LED, displaying output on 16x2 display, Buzzer (in collision), and phone notification in case collision occurs.

   - Input Devices: Ultrasonic Sensor
   - Output Devices: LEDs, 16x2 Display, Buzzer, Smartphone, Arduino Bluetooth module, micro servo.

3. **Communication:** The system implements a simple-yet-effective communication mechanism which enables a variety of features as our system contains different types of output devices.

   - As the program starts, the ultrasonic sensor starts reading distances in its field of view immediately. At this point, the data is being fed to the program, which outputs the distance in centimeters to the 16x2 display. The "distance range" is then figured out by multiple if statements and specific LEDs indicate the distance range of the vehicle.
   - In case a collision occurs, the buzzer will sound and at that moment a notification will be sent to the user's phone connected via Bluetooth to the Arduino (courtesy

of the Bluetooth module) notifying them of the occurrence of said collision. At this point, the array of LEDs will also be completely lit up as a visual queue that a collision has occurred.

4. **Original Work:** The original work of this project includes measuring distance of a specific object and lighting up a specific colored LED according to the distance of the object. We are taking the same idea and adding a variety of output devices to the system which will bring a variety of new and advanced functionality to the original system, such as convenience of the smartphone to monitor driving habits, finding out actual distance between the sensor and the object, alerts in case of collision, etc.

Supporting Materials:

1. **Timeline:**

- 10/9/20 – Milestone 3
- 10/16/20 – Completed circuit design with potential improvements and additions, order additional hardware
- 10/23/20 – Finalized circuit design and tested individual components for checking functionality
- 10/30/20 – Finished building actual circuit for the project and partially started writing code for project functionality
- 11/6/20 – Started handling issues (hardware and software), and submitted updated design document
- 11/13/20 – Completed the project with proper functionality

- 11/20/20 – Fixed last-minute issues and worked on project video
- 11/23/20 – Project Presentation
- 12/04/20 – Submitted final design document

2. **Materials Required:** Smartphone, Arduino Bluetooth Module, Buzzer, LEDs, Ultrasonic Sensor, 16x2 display, Micro Servo, Arduino Board, Bread board, resistors, connecting wires.
3. **References:**

- **Bluetooth:** This article shows us the basics of how a Bluetooth module can be used to control the state of an LED using an Android app.

  Arduino Bluetooth Basic Tutorial. (2016, May 23). Arduino Project Hub. https://create.arduino.cc/projecthub/mayooghgirish/arduino-bluetooth-basic-tutorial-d8b737

- **Micro Servo:** This article shows us how to setup and rotate the micro servo.

  Working with a Micro Servo. (2019, April 25). Arduino Project Hub. https://create.arduino.cc/projecthub/sumanbargavr/working-with-a-micro-servo-86ec6b

- **Buzzer:** This article shows us how to use a piezo buzzer to emit an audible sound.

  USE a BUZZER MODULE (PIEZO SPEAKER) USING ARDUINO UNO. (2018, June 5). Arduino Project Hub. https://create.arduino.cc/projecthub/SURYATEJA/use-a-buzzer-module-piezo-speaker-using-arduino-uno-89df45

- **Original Project:** This article shows us how we can use and ultrasonic sensor to measure the distance between said sensor and an object.

  How to Use an Ultrasonic Sensor. (2018, December 2). Arduino Project Hub. https://create.arduino.cc/projecthub/MisterBotBreak/how-to-use-an-ultrasonic-sensor-181cee

4. **Hardware Diagrams:**

   https://github.com/sshekh7/CS362Project/blob/main/cs362_hardware_diagram_complete.pdf

5. **Software Diagrams:**

   https://github.com/sshekh7/CS362Project/blob/main/cs_362_code.ino

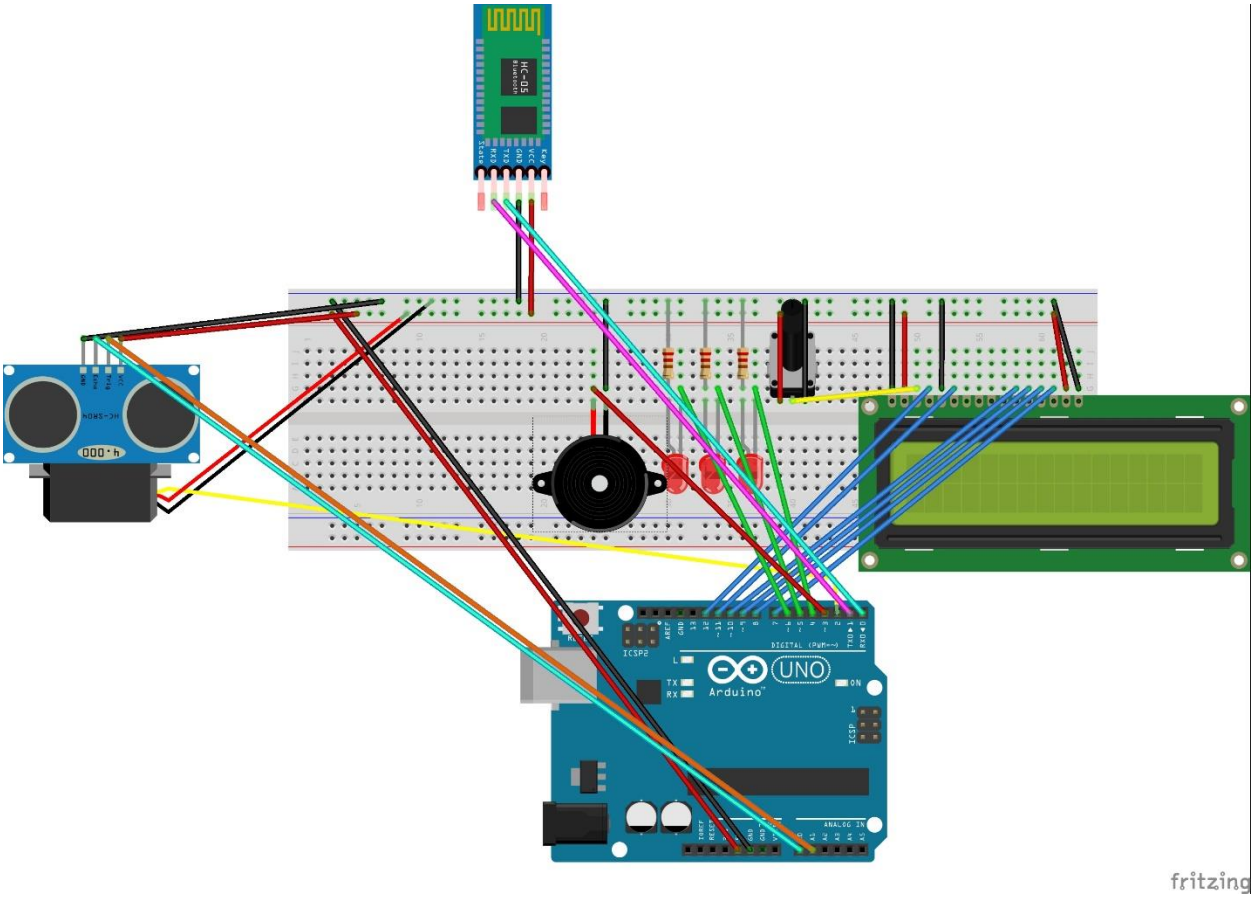
Required Code:

```
#include <LiquidCrystal.h>
#include <Servo.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int green_led = 6;
int yellow_led = 7;
int red_led = 8;

Servo servo;

int buzzer = 13;

// ultrasonic sensor
const int trig = 9;
const int echo = 10;

// for keeping track of the distance
int duration = 0;
int distance = 0;

void setup() {
  lcd.begin(16,2);
  pinMode(green_led, OUTPUT);
  pinMode(yellow_led, OUTPUT);
  pinMode(red_led, OUTPUT);
  pinMode(trig , OUTPUT);
  pinMode(echo , INPUT);
  Serial.begin(9600);
  servo.attach(A0);
}

void loop() {
  lcd.clear();
  lcd.setCursor(0,0);
  digitalWrite(trig , HIGH);
  digitalWrite(trig , LOW);

  duration = pulseIn(echo , HIGH);
  distance = (duration/2) / 28.5 ;

  if( distance < 0){
```

```
    Serial.print("d");
    delay(300);
    for(int i=0;i<5;i++)
    {
      tone(buzzer, 1100);
      delay(500);
      tone(buzzer, LOW, 500);
      delay(150);
    }
    exit(0);
    // the system has to be restarted manually by the car mechanic in case of an
accident.
  }

  if ( distance <= 7 )
  {
    digitalWrite(red_led, HIGH);
    lcd.setCursor(0,1);
    lcd.print("WARNING");
  }
  else {
    digitalWrite(red_led, LOW);
  }


  if ( distance <= 15 )
  {
    digitalWrite(yellow_led, HIGH);
    lcd.setCursor(0,1);
    lcd.print("KEEP DISTANCE");
  }
   else
  {
    digitalWrite(yellow_led, LOW);
  }

  if ( distance <= 25 )
  {
    digitalWrite(green_led, HIGH);
    lcd.setCursor(0,1);
    lcd.print("SAFE");
  }
  else
  {
    digitalWrite(green_led, LOW);
```

```
    Serial.print("n");
    delay(300);
  }
  lcd.setCursor(0,0);
  lcd.print("Distance: ");
  lcd.print(distance);


  servo.write(0);
  delay(500);
  servo.write(90);
  delay(500);
  servo.write(180);
  delay(500);
  servo.write(90);
  delay(500);
  servo.write(0);
  delay(500);
}
```

Hardware Diagram:



How to build project:

- ➔ **Basic Requirements**: Arduino UNO Board, Breadboard, Connecting wires, 220 Ω resistors, Hot-glue gun (or double-sided tape).
- ➔ **Required I/O parts**: Ultrasonic Sensor, Micro Servo, LEDs, 16x2 Display, Buzzer, Smartphone, Arduino Bluetooth module.
- ➔ **Circuit Design:**
  - **LCD:**
    - Connect pins 1, 5, 16 to the negative terminal of the bread board.
    - Connect pins 2, 15 to the positive terminal of the bread board.
    - Connect pin 3 to the middle pine of the potentiometer.
    - Connect pin 4 to pin 12 of the Arduino board.
    - Connect pin 6 to pin 11 of the Arduino board.
    - Connect pin 11, 12, 13, 14 to pins 10, 9, 8, 7 respectively, on the Arduino board
  - **Potentiometer:**
    - Connect one end of the potentiometer to the negative terminal of the bread board
    - Connect the other end to the positive terminal.
    - Connect the middle pin to pin 3 of the LCD (as stated in LCD connection).
  - **LEDs:**
    - Connect one end of a 220-ohm resistor to the negative terminal of the bread board.
    - Connect the other end of the 220-ohm resistor to the negative end of an LED.
    - Repeat the above steps for the remaining LEDs.

- Connect the positive end of the each of the three LEDs to pins 6, 5, 4 of the Arduino board respectively.

- **Active Buzzer:**
  - Connect negative end of the buzzer to the negative terminal of the bread board.
  - Connect the positive end of the buzzer to pin 3 on the Arduino board.
- **Micro Servo:**
  - Connect the black wire (negative terminal) of the micro servo to the negative terminal of the bread board.
  - Connect the red wire (positive terminal) of the micro servo to the positive terminal of the bread board.
  - Connect the yellow wire of the micro servo to pin 2 on the Arduino board.
- **Ultrasonic Sensor:**
  - Connect the VCC end to the positive terminal of the bread board.
  - Connect the GND end to the negative terminal of the bread board.
  - Connect the ECHO end to the pin A0 (utilized analog pin as digital pin, as we ran out of normal digital pins).
  - Connect the TRIG end to the pin A1 (utilized analog pin as digital pin, as we ran out of normal digital pins).
- **Bluetooth Module (hc-06/hc-05):**
  - Connect the VCC end to the positive terminal of the bread board.
  - Connect the GND end to the negative terminal of the bread board.
  - Connect TXD to RXD (pin 0) on the Arduino board.
  - Connect RXD to TXD (pin 1) on the Arduino board.

- **Other:**
    - Connect the 5V pin of the Arduino board to the positive terminal of the bread board.
    - Connect the GND pin of the Arduino board to the negative terminal of the bread board.
    - Mount the ultrasonic sensor on the micro servo (to allow rotation) using a hot glue gun.

How project is used:

The Car Collison Detection System utilizes an Ultrasonic sensor as a radar to scan nearby surroundings for potential hazards/unknown objects. If an object is detected in the proximity of the range of the Ultrasonic sensor, a blip is shown on the radar screen and the distance to the object is shown on the LCD display. For a car in movement, the LED's indicate the distance of nearby objects (possibly, other vehicles). The more LED's that are lit, the closer the object is. The frequency of the buzzer increases as the object gets closer and reaches its maximum on collision with another object, and additionally, any smartphone connected via Bluetooth to the system, is notified instantly about the collision.

If the system is to be utilized on an actual vehicle, some (very slight) modifications are required in the system. These changes include:

- Using a slightly better ultrasonic sensor with a longer range to read data from. The ultrasonic sensor used in the system is a short-range sensor which is capable of detecting objects from a short range of length. Additionally, the current ultrasonic sensor is buggy

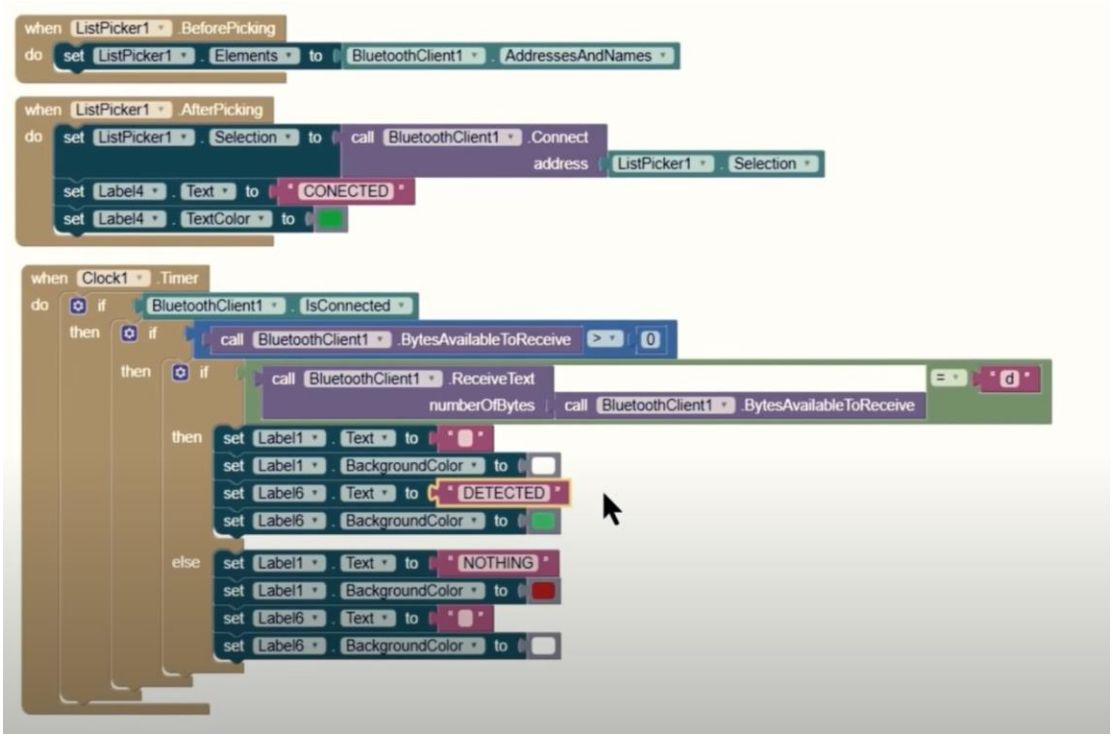
in reading distances. In real life situations, **multiple** long range and accurate ultrasonic sensors with very low latency should be attached to the vehicle at different places in the vehicle including the front, the back, and both the sides of the car to detect collisions, and accurately measure distances of the other vehicles.

- The Arduino UNO used in the projects does not comes with any inbuilt networking card, or wireless network support. To support wireless communication of the system with a secondary device, a Bluetooth module is utilized (as it was the cheapest and easiest way to support this facility). An actual implementation in a vehicle of the system, can implement an Arduino MKR GSM1400 board, as it is capable of sending actual text messages to smartphones (or any mobile phone in the GSM network bands), which makes sense as a customer may not always be super close to the vehicle to be in the Bluetooth connectivity range of the system.
- To get a 180° field of view for the ultrasonic sensor, the sensor is mounted on top of a micro servo which rotates a full 180° to support this functionality. In an actual implementation, multiple ultrasonic sensors, or a sensor with a compact inbuilt rotating mechanism can be utilized to achieve the same functionality.
- The LED indicators are signals for quick-peeking the car's distance from other vehicles, while the real-time distance on the 16x2 display is for testing purposes. An actual implementation will have the real-time distance, and the LED indicators inbuilt in the rear-view mirror of the car. This will allow the driver to see real-time distance in the back of the car while moving the car in reverse.

- The buzzer is the clearest indicator that a collision has occurred (the driver has been hit/or the driver has hit), which should be replaced with the alarm system of the vehicle in an actual implementation.

Problems Faced (and how we solved them)

- The biggest problem we faced while adding the Bluetooth functionality to the system was finding out a way/method to read data from the Bluetooth module in the system. The "MIT App Inventor" is a software we utilized to build an android application which can successfully read data from the Bluetooth module. Below is the schematic of how the app works:



If a collision is detected the following line of code executed:

```
Serial.print("d");
```

The Bluetooth module send this data to the application, which changes the color of the button to green from red.

- Limited area of working for the ultrasonic sensor was the next big problem. Look, the ultrasonic sensor is static, and the HC-SR04 model does not contains an inbuilt motor to rotate the sensor so that it can read values from wider range of area. With simply using the sensor as it is, it will read values just in front of the sensor. Although an actual implementation should utilize multiple sensors for a quick-yet-effective reading of data, we mounted the ultrasonic sensor on top of a micro servo. The below image shows our current implementation of the setup.

The following code from the program:

```
servo.write(0);
delay(500);
servo.write(90);
delay(500);
servo.write(180);
delay(500);
servo.write(90);
delay(500);
servo.write(0);
delay(500);
```

These lines of code rotate the ultrasonic sensor 0-90-180-90-0 degrees. The implementation expands the view of the sensor and we get a 180° area for reading the distances using the sensor.