GROUP ASSIGNMENT
(INDIVIDUAL WORK)

CT101-3-1-IIOT-L-2

INTRODUCTION TO INTERNET OF THINGS

APD1F2309IT

NAME : ELEANOR PERMATA FRY

STUDENT ID : TP072606

ASSIGNMENT : ARDUINO MAKER UNO REVERSE CAR PARK SYSTEM

LECTURER : DR KAMALANATHAN SHANMUGAM

HAND OUT DATE : 8 MAY 2024

HAND IN DATE : 13 JUNE 2024

WEIGHTAGE : 50%

Table of Contents

1.0 Introduction

1.1 Introduction to Internet of Things (IoT)

The Internet of Things (IoT) is a revolutionary idea that describes a network of interconnected physical items, sensors, software, and technologies that communicate and share data through the Internet or other communication networks. This networked system allows "smart" devices to examine, process, and communicate data, resulting in a wide range of applications across numerous sectors (Mahmudova, 2021). The term "Internet of Things" was first used in 1999 by British entrepreneur and startup founder Kevin Ashton, who is best known for co-founding the Auto-ID Laboratory at MIT (Rejeb et al., 2019).

Figure 1 – Kevin Ashton

Kevin Ashton's early work in presenting the notion of IoT paved the way for the continued development and expansion of this revolutionary technology. His concept of connecting the physical world to the internet through breakthrough technologies like as radio-frequency identification (RFID) tags laid the groundwork for today's interconnected environment (Tiruvayipati, 2024). Ashton's efforts have altered the IoT landscape while also inspiring breakthroughs in supply chain management, healthcare, smart cities, and industrial automation (Gaitan, 2021).

As IoT evolves and integrates into more aspects of our life, Ashton's first observations about the possibility of connecting everyday devices to the internet become increasingly relevant. The IoT ecosystem, with its massive network of networked devices, has huge potential for increasing efficiency, productivity, and connectivity across businesses and sectors (Korte et al., 2021). Ashton's foundational work demonstrates the revolutionary power of IoT and its promise to change the way we interact with technology and the environment around us.

1.2 Introduction to Arduino Maker UNO

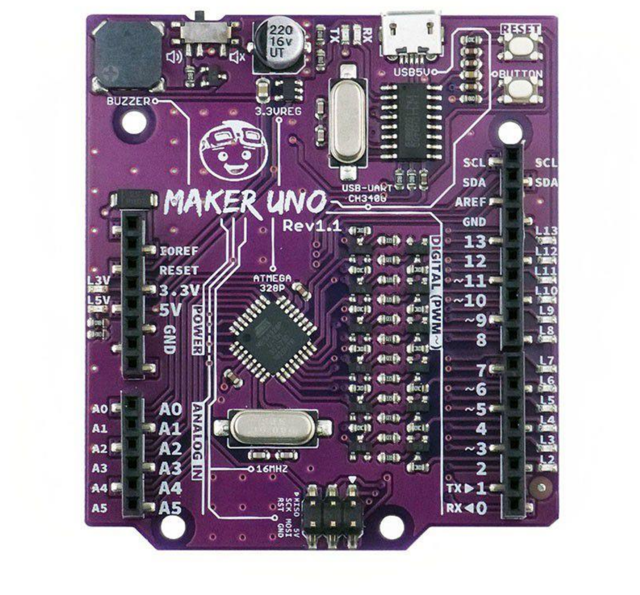

Figure 2 – Maker UNO

The Maker UNO, a microcontroller board based on the ATmega328P microprocessor, was designed to be a versatile and cost-effective tool for numerous applications (Wei, et al., 2021). It provides a user-friendly environment for both beginner and advanced users to work on electronics projects, automation, and programming. The Maker UNO, which is compatible with Arduino software and a variety of sensors and actuators, offers a practical and accessible approach to learn about electronics, coding, and automation (Rahman & Hossain, 2021).

The Maker UNO's simplicity and ease of use make it an excellent educational tool, allowing students to learn the principles of electronics and programming through hands-on experience (Firouzi et al., 2022). Its interaction with the Arduino environment enables a wide range of projects to be done, from simple LED blinking exercises to more complicated automation and IoT applications. The Maker UNO acts as a doorway for people to explore the worlds of electronics, robotics, and IoT, encouraging creativity and innovation in the field of technology (Tsang et al., 2021).

Furthermore, the Maker UNO's interaction with the Internet of Things (IoT) environment brings up an endless number of opportunities. IoT allows seamless device communication and data exchange, enabling for remote system control and monitoring (Romeo et al., 2020). The integration of IoT with microcontroller boards like as the Maker UNO has accelerated the development of smart applications in fields such as smart homes, agriculture, and healthcare (Priyadarshini et al., 2022).

1.3 Overview of the Reverse Car Park System

1.3.1 Introduction

With the growing number of vehicles and lack of parking spaces, efficient parking systems are needed to maximize the use of available space and guarantee safety. Reverse parking assistance are an essential component of modern parking systems; they help drivers park their cars safely by identifying obstructions behind their vehicles. In order to assist drivers when reversing their cars, this project will create and build a reverse car park system which makes use of the Arduino Maker UNO to provide audio and visual of when an obstacle is near as well as their proximity to the reversing vehicle.

Figure 3 – Visual Idea of Reverse Car Park System in Real-Life Situation

1.3.2 Aims and Objective

The main objective of this project is to use the Arduino Maker UNO to build a functional reverse car parking system. When an obstacle is detected behind the car, the system will notify the driver using buzzer and LEDs. This project is to improve the understanding of sensor integration, signal processing and user interface design while showcasing the usage of Arduino in practical car safety system.

2.0 System Design

2.1 Project Components

This project utilizes a handful of components which all serves different purposes once it runs, those components include an Arduino Maker UNO, ultrasonic sensor, LEDs, buzzer, breadboard, wires, power supply, and resistors. Here are the explanations for all the components used and their functional:

1. Arduino Maker UNO

The main microcontroller used in this project is the Arduino Maker UNO. It analyzes the ultrasonic sensor's input data and manages the output devices, including the buzzer and LEDs. In order to notify the driver of incoming obstacle, the Maker UNO is designed to read sensor data, make assumptions based on the values received, and then activate the proper feedback systems.

2. Ultrasonic Sensor (HC-SR04)

The distance between the car and obstacles behind is measured using the HC-SR04 ultrasonic sensor. It generates ultrasonic waves and measures how long it takes to return after colliding with an object. The distance is measured and then delivered to the Arduino for processing. The sensor gives the system the distance information continually, which allows it to calculate how close obstacles are.

3. 16x2 LCD

The 16x2 LCD panel is used to display real-time distance measurements and status messages. It gives the driver easy-to-read interface, displaying precise distance to obstacles and signaling if the conditions are safe, cautionary or dangerous. By offering comprehensive information to add to the visual and audio inputs, the LCD improves the system.

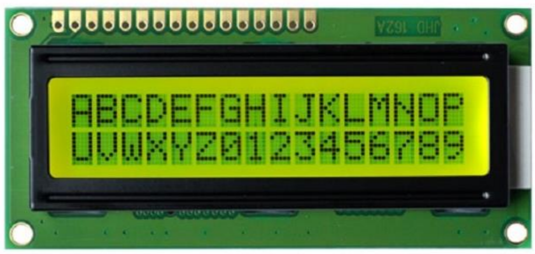

Figure 4 – 16x2 LCD

4. LEDs (Red, Yellow, Green)

LEDs are an electronic device, which emits light when a current flows through its terminals. In this project, LEDs are used to indicate the distance between the obstacle and the vehicle. When the LED turns green, it means that there are no obstacles around the vehicle (more than 10 cm). When the LED turns yellow, it means that the vehicle is approaching an obstacle (distance is between 5 to 10 cm). On the other hand, the LED will turn red when the obstacle is very close to the vehicle (less than 5 cm). The driver may quickly visualize the range of the obstacles thanks to these LEDs.

5. Piezoelectric Buzzer

When the vehicle is getting close to an obstacle, the buzzer alerts the driver by making noise. As the distance to the obstacles decreases, the beep frequencies increases, providing a clear indication of growing approach. When the driver is not paying attention to the visual signs, this audio alert is especially helpful.

Figure 5 – Piezoelectric Buzzer

6. Breadboard and Jumper Wires

All the circuit's components are easily connected and prototyped thanks to the breadboard and jumper wires. This configuration is ideal for development and troubleshooting since it makes it simple to test and modify circuit connections.

7. Power Supply

The Arduino Maker UNO and other components receive the electrical power the require from a power supply. In this case a USB connection from a computer is used as a power supply, though a battery pack and a power bank can also be used for power. Maintaining a steady power supply is essential to the system's reliable operation.

8. Resistors

In this circuit a 220 ohm resistor is used for the buzzer, while the LEDs makes use of the 1 kiloohm. Resistors are used to keep the LEDs from burning out, resistors are used to control the amount that lows to them. Resistors add to them system's durability and dependability by making sure each LED runs within its safe current range.

2.2 Tinkercad Model

The figure below shows an overview of how the project is made on Tinkercad and how each component is connected as well as the functions within the system. The virtual environment helps to test the functioning system and troubleshoot any faults before configuring it with real components.

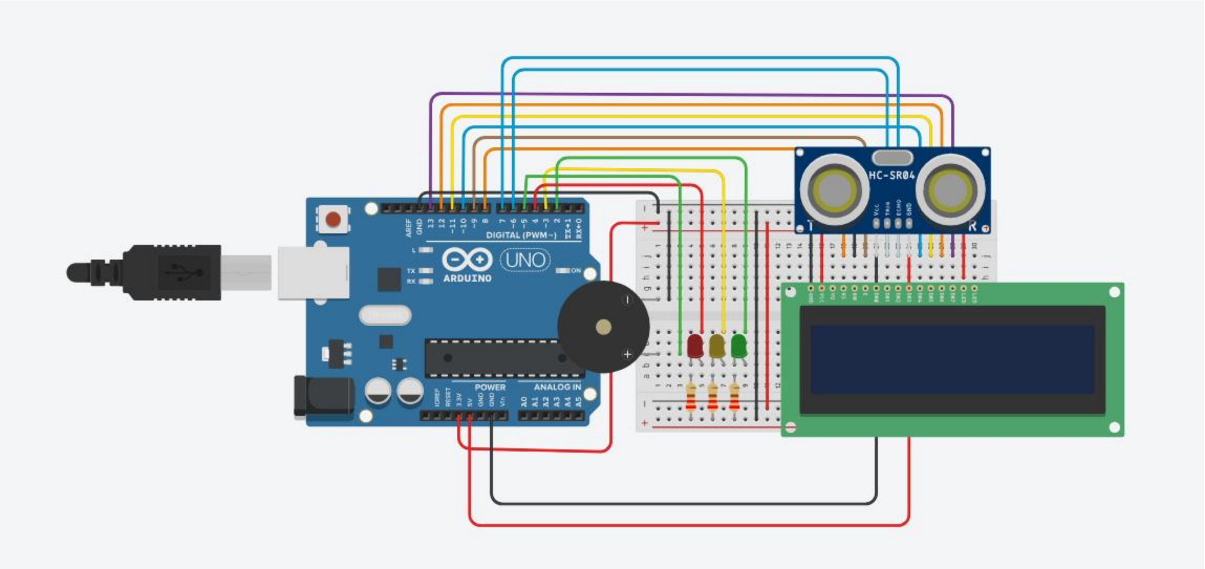

Figure 6 – Tinkercad Model

3.0 Prototype Screenshot

3.1 Physical Setup Overview

The figures below show the physical setup of the reverse car parking system, showcasing the connections of all the components used in the project.

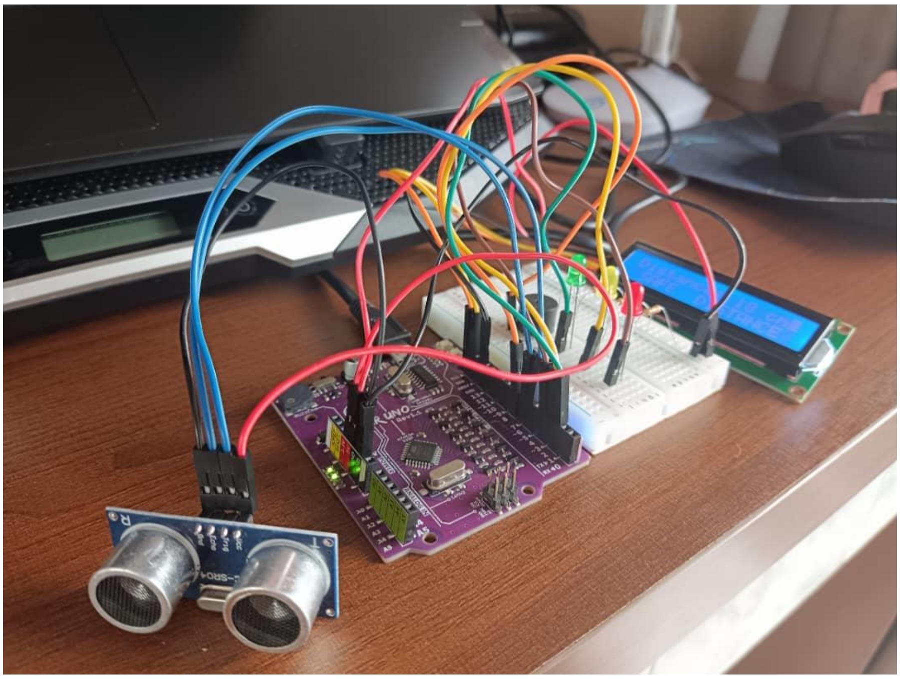

Figure 7 – Physical Setup (Side View)

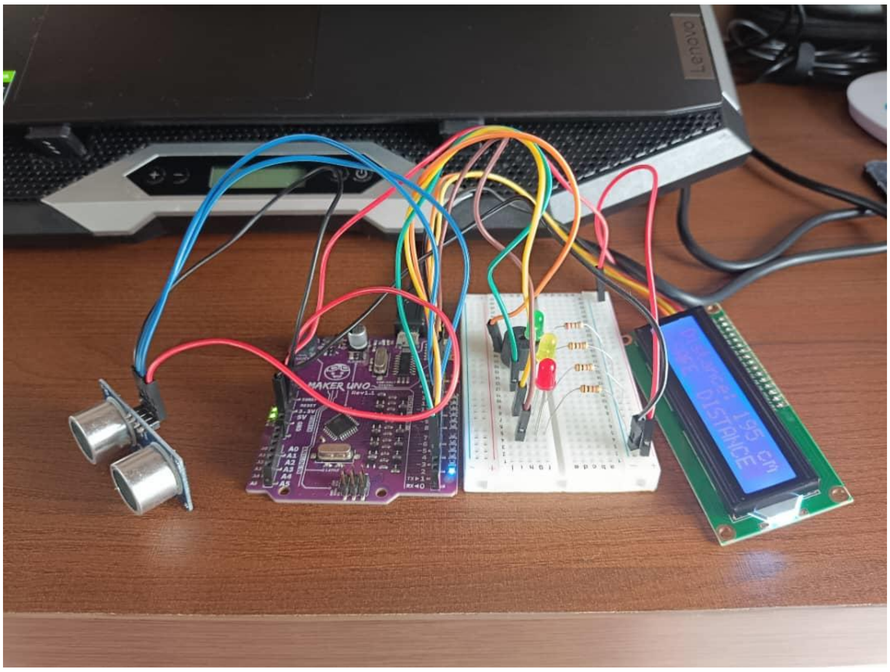

Figure 8 – Physical Setup (Top View)

3.2 Assembly Instructions

Here are the instructions on how to assemble the physical setup of the project.

1. Setup the Arduino Maker UNO and the breadboard. Connect all the LEDs, buzzer, and resistors on to the breadboard.
2. Use the wires to connect the supply and ground from the Arduino to the breadboard.
3. Connect the 1 kiloohm to each cathode (short leg) of the LEDs and the positive legs of the LEDs to the designated port of the Arduino using the wires, where the green LED would head to digital pin 2, yellow LED will head to digital pin 3, and red LED will head to pin 4 on the Arduino

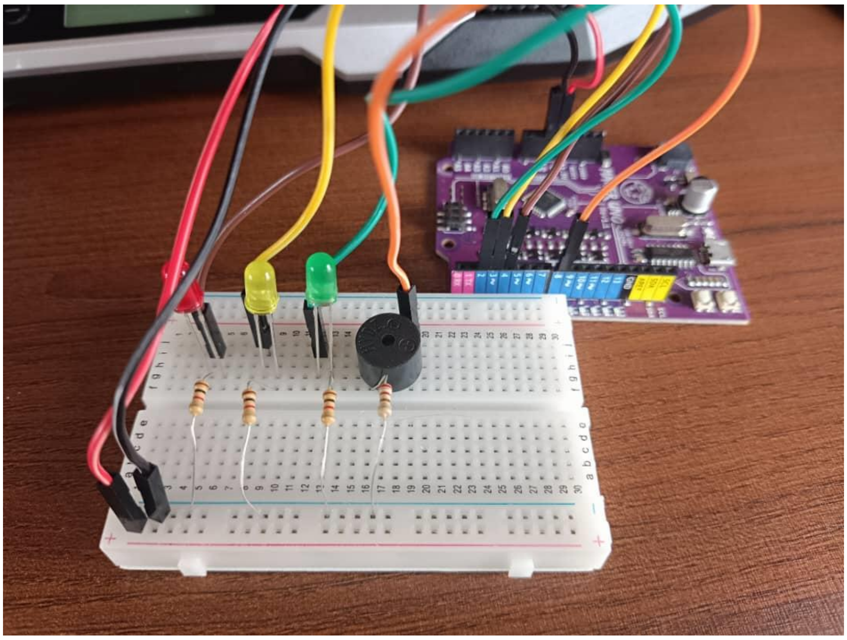

Figure 9 – Connect the LEDs, Buzzer, and Resistors

4. Connect the positive leg of the buzzer to digital pin 8 of the Arduino and the negative led of the buzzer to the 220-ohm resistors which is then connected to the GND rail on the breadboard.
5. Connect the LCD screen to the Arduino and breadboard. Connect the VCC pin of the LCD to the power supply rail of the breadboard, GND pin to the GND pin, SDA pin to the SDA pin and SCL pin to the SCL pin on the Arduino.

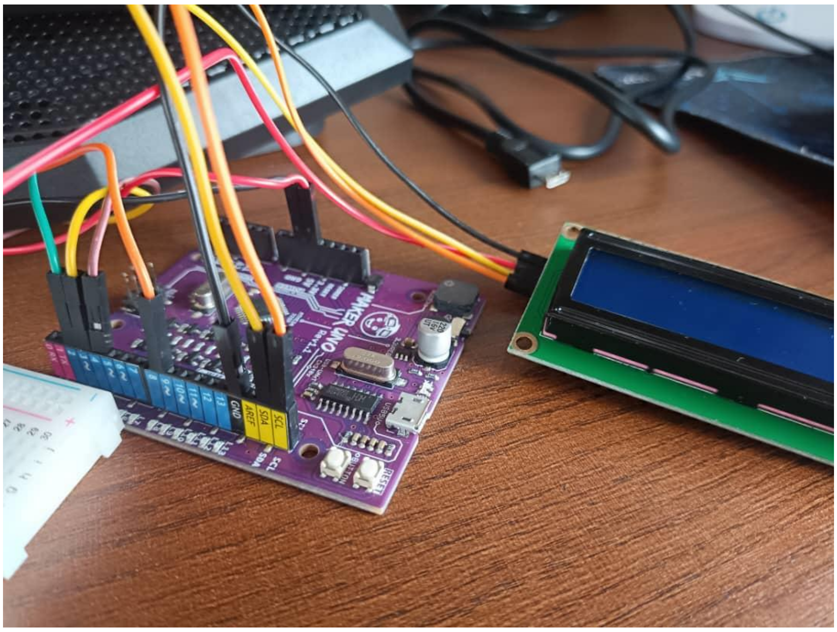

Figure 10 – Connect the LCD

6. Connect the ultrasonic sensor (HC-SR04) to the Arduino. Connect the VCC pin of the HC-SR04 to the 5V pin, GND pin to the GND pin, Trig pin to digital pin 7, and Echo pin to digital pin 6 on the Arduino.

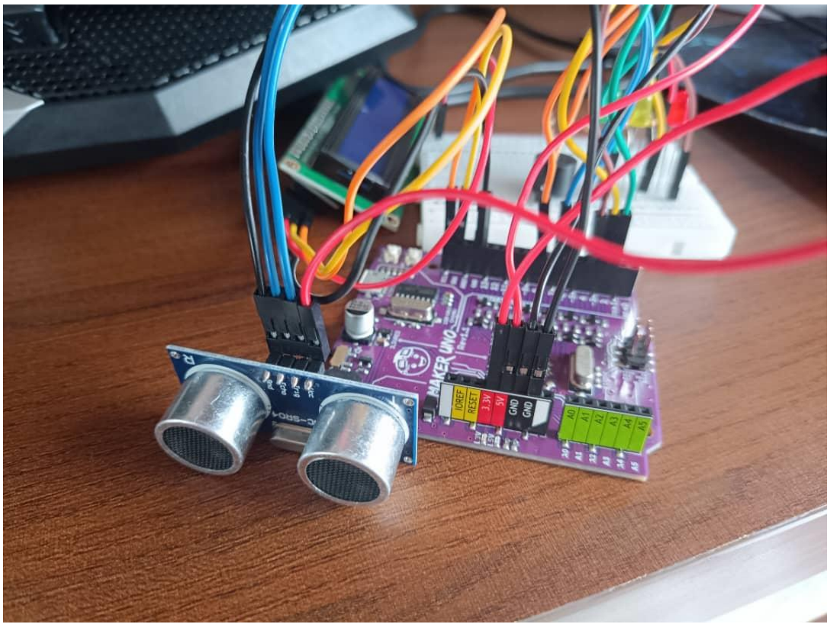

Figure 11 – Connect the Ultrasonic Sensor

7. Once done, connect the Arduino to a power supply using a USB cable that connects to the computer.
8. Lastly, upload run the code in Arduino IDE.

3.3 System Manual

Here are the manuals on how to use the Reverse Car Parking System.

1. Make sure that all the components are properly connected to the Arduino. Once the code is running, make sure that there are no problems or mistakes.
2. Use any objects to test the system. Place the object within the range of the sensor.
3. When an object is in range of the sensor within 5 to 10 cm, the yellow LED will blink and the buzzer will make a noise continuously. The LCD should display the proper distance and a warning notification to the driver/user.
4. When the object in range in within 4 cm or less, the red LED should light up and the buzzer will make a continuous tone to notify the driver/user that the object is getting very close to the vehicle. The LCD should display the proper distance as well as a danger notification.
5. If there are no objects in range or the distance is above 10 cm, the green LED will light up, no sound will be emitted from the buzzer and a safe notification will be displayed on the LCD.

4.0 Source Code

4.1 Complete Source Code

The text below provides the source code for the Arduino Maker UNO-based reverse car park system. The source code utilizes several functions and key components to measure the distance and range of an obstacle and provide visual and auditory feedback towards the user or driver of the vehicle.

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

const int buzzer = 8;

int pinGREEN = 2;
int pinYELLOW = 3;
int pinRED = 4;

int echoPin = 6;
int trigPin = 7;

int duration = 0;
int distanceCM = 0;

void setup() {
  lcd.init();
  lcd.backlight();

  pinMode(buzzer, OUTPUT); // Buzzer
  pinMode(pinGREEN, OUTPUT); // Green LED
  pinMode(pinYELLOW, OUTPUT); // Yellow LED
  pinMode(pinRED, OUTPUT); // Red LED

  pinMode(echoPin, INPUT); // Ultrasonic Sensor
  pinMode(trigPin, OUTPUT);
  Serial.begin(9600);

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distanceCM = duration * 0.017175;
```

```
Serial.print("Distance: ");
Serial.print(distanceCM);
Serial.println(" cm");

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Distance: ");
lcd.print(distanceCM);
lcd.println(" cm");

if (distanceCM <= 4) {
  digitalWrite(pinRED, HIGH);
  digitalWrite(pinYELLOW, LOW);
  digitalWrite(pinGREEN, LOW);
  tone(buzzer, 2000);

  lcd.setCursor(0, 1);
  lcd.print("DANGER DISTANCE");
}
else if (distanceCM <= 10 && distanceCM >= 5) {
  digitalWrite(pinRED, LOW);
  digitalWrite(pinYELLOW, HIGH);
  digitalWrite(pinGREEN, LOW);
  noTone(buzzer);

  for (int i = 0; i < 5; i++) {
    digitalWrite(pinYELLOW, HIGH);
    tone(buzzer, 2000);
    delay(100);
    digitalWrite(pinYELLOW, LOW);
    noTone(buzzer);
    delay(100);

    lcd.setCursor(0, 1);
    lcd.print("WARNING DISTANCE ");
  }
}
else {
  digitalWrite(pinRED, LOW);
  digitalWrite(pinYELLOW, LOW);
  digitalWrite(pinGREEN, HIGH);
  noTone(buzzer);

  for (int i = 0; i < 5; i++) {
    digitalWrite(pinGREEN, HIGH);
    delay(300);
    digitalWrite(pinGREEN, LOW);
```

```
      delay(300);

      lcd.setCursor(1, 1);
      lcd.print("SAFE  DISTANCE");
    }
  }
  delay(50);
}
```

4.2 Include Libraries and Initialize Components

In the beginning of the system, it includes the LiquidCrystal_I2C library and initializes an LCD object with a 16x2 character display by setting the I2C address to 0x27, which is used for controlling the LCD displays that uses the I2C protocol to communicate. With the help of this library, connecting an LCD screen to an Arduino is done with ease, allowing text and numbers to be displayed without needing the complicated details of the communication protocol.

Then in the next part of the setup includes by defining the pin numbers for the ultrasonic sensor's echo and trigger pins, as well as those for the buzzer (8), green (2), yellow (3), and red (4) LEDs. Furthermore, variables called duration and distanceCM are also declared to capture the time length of the echo signal and the computed distance.

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

const int buzzer = 8;

int pinGREEN = 2;
int pinYELLOW = 3;
int pinRED = 4;

int echoPin = 6;
int trigPin = 7;

int duration = 0;
int distanceCM = 0;
```

4.3 Setup Function

In the 'setup()' function, the first thing that happens is that the LCD is initialized and its backlight is turned on. The ultrasonic sensor pins, LEDs, and buzzer may all be set to be either input or output depending on the situation using the pinMode() function. For debugging purposes, serial communication is also started at a baud rate of 9600. The loop() method, which is called repeatedly, contains the program's fundamental logic.

```
void setup() {
  lcd.init();
  lcd.backlight();

  pinMode(buzzer, OUTPUT); // Buzzer
  pinMode(pinGREEN, OUTPUT); // Green LED
  pinMode(pinYELLOW, OUTPUT); // Yellow LED
  pinMode(pinRED, OUTPUT); // Red LED

  pinMode(echoPin, INPUT); // Ultrasonic Sensor
  pinMode(trigPin, OUTPUT);
  Serial.begin(9600);
}
```

4.3 Main Loop

The 'loop()' function starts by sending a 10-microsend pulse to the ultrasonic sensor's trigger pin to start measuring the distance. The echo pulse's duration is then measured, and the distance is computed through the sound speed. The distance is displayed on the LCD panel and written to the Serial Monitor. The system gives the driver different feedback according to the measured distance.

```
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distanceCM = duration * 0.017175;

  Serial.print("Distance: ");
  Serial.print(distanceCM);
  Serial.println(" cm");

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Distance: ");
  lcd.print(distanceCM);
  lcd.println(" cm");
```

4.5 Distance-Based Actions

In this segment, an if-else statement is used to creates actions based on the distance between the sensor and the obstacle. If the distance happens to be less than or equal to 4 cm, the red LED will turn on and the buzzer will continuously emit a 2000 Hz tone. On the LCD panel, the distance is measured on the first line, while the second line will display “DANGER DISTANCE” to warn the driver of the impending danger. In the else-if statement, the distance measured is between 5 to 10 cm, both the yellow LED and the buzzer will blink and beep continuously. The LCD panel displays the message “WARNING DISTANCE” at the second row of the screen. When the distance is larger than 10 cm, the buzzer is disabled and the green LED will turn on and blink continuously. This scenario indicates a safe distance between the driver and any obstacles, “SAFE DISTANCE” is displayed on the LCD panel.

The loop() function includes a 50 millisecond wait at the end to ensure stable functioning and prevent any excessive sensor readings. By using a complete approach, safety and convenience are increased as the driver is guaranteed to receive input in real time regarding the closeness of any obstacles when reversing.

The code efficiently illustrates how to combine a handful of parts—including the buzzer, LCD, LEDs, and ultrasonic sensor—to use the Arduino Maker Uno to build a useful and working reverse parking system.

```
if (distanceCM <= 4) {
   digitalWrite(pinRED, HIGH);
   digitalWrite(pinYELLOW, LOW);
   digitalWrite(pinGREEN, LOW);
   tone(buzzer, 2000);

   lcd.setCursor(0, 1);
   lcd.print("DANGER DISTANCE");
 }
 else if (distanceCM <= 10 && distanceCM >= 5) {
   digitalWrite(pinRED, LOW);
   digitalWrite(pinYELLOW, HIGH);
   digitalWrite(pinGREEN, LOW);
   noTone(buzzer);

   for (int i = 0; i < 5; i++) {
     digitalWrite(pinYELLOW, HIGH);
     tone(buzzer, 2000);
     delay(100);
     digitalWrite(pinYELLOW, LOW);
```

```
      noTone(buzzer);
      delay(100);

      lcd.setCursor(0, 1);
      lcd.print("WARNING DISTANCE ");
    }
  }
  else {
    digitalWrite(pinRED, LOW);
    digitalWrite(pinYELLOW, LOW);
    digitalWrite(pinGREEN, HIGH);
    noTone(buzzer);

    for (int i = 0; i < 5; i++) {
      digitalWrite(pinGREEN, HIGH);
      delay(300);
      digitalWrite(pinGREEN, LOW);
      delay(300);

      lcd.setCursor(1, 1);
      lcd.print("SAFE  DISTANCE");
    }
  }
  delay(50);
}
```

5.0 Limitation

While the Arduino Maker UNO Reverse Car Park System offers a useful way to improve car and the driver's safety, it won't be able to properly do so without its limits. First, even though the ultrasonic sensor works well for obstacle detection, external conditions like dust, rain, or extremely high temperatures can possibly damage it in the long-run and cause readings to be off. Second, there may be false alerts since the system simply measures distance and ignores the size, shape, or type of obstacle. Although helpful, the simple LED and buzzer feedback system might not be suitable for all drivers, particularly in busy environments where the buzzer's alert might be muffled. Lastly, the system has small delays that could hinder quick responses, which makes it less useful in situations where quick feedback is required.

6.0 Future Enhancement

To overcome current restrictions an limitations of the reverse parking system, I can take into consideration a number of future enhancements to improve the functionalities of the system. Combining several sensors—like radar or infrared—could greatly improve accuracy and reliability in a range of environmental circumstances. The development of a more advanced alert system that include voice alerts or visual displays would give the driver access to more specific information. Furthermore, by adding machine learning techniques, the system might be better to be able to identify various anticipations of possible collisions. Adding wireless connection capabilities to the system would enable real-time data transfer to a smartphone app, providing drivers with a more feature-rich and intuitive interface. The reverse parking system would be more reliable, accurate, and user-friendly with these improvements.

7.0 Conclusion

Having the opportunity to work on this project has been an enlightening experience as it has demonstrated the real-world application of Arduino Maker UNO in creating a reverse parking system. The project I've created provides an affordable way to help drivers to avoid any obstacles when reversing. With the help of LEDs, a buzzer, and ultrasonic sensors, I was able to effectively develop a system that improves safety by giving real-time proximity feedback. Despite the challenges and limitations faced during the making of this project, I have learned a great deal about sensor interface, signal processing, and user interface design through this project, laying the foundation for future research into embedded systems and automotive electronics.

8.0 References

Firouzi, F., Farahani, B., & Marinšek, A. (2022). The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT). *Information Systems*, *107*, 101840.

Gaitan, N. C. (2021). A Long-Distance Communication Architecture for Medical Devices Based on LoRaWAN Protocol. *Electronics, 10*(8), 940.

Korte, A., Tiberius, V., & Brem, A. (2021). Internet of Things (IoT) Technology Research in Business and Management Literature: Results from a Co-Citation Analysis. *Journal of Theoretical and Applied Electronic Commerce Research*, *16*(6), 2073-2090.

Mahmudova, S. (2021). Intelligent systems for the internet of things: essence, perspectives and problems. *International Journal of Natural Sciences Research, 9*(1), 17-25.

Priyadarshini, I., Bhola, B., Kumar, R., & So-In, C. (2022). A Novel Cloud Architecture for Internet of Space Things (IoST). *IEEE Access, 10*, 15118-15134.

Rahman, M. A., & Hossain, M. S. (2021). An Internet-of-Medical-Things-Enabled Edge Computing Framework for Tackling COVID-19. *IEEE Internet of Things Journal, 8*(21), 15847-15854.

Rejeb, A., Keogh, J., & Treiblmaier, H. (2019). Leveraging the Internet of Things and Blockchain Technology in Supply Chain Management. *Future Internet*, *11*(7), 161.

Romeo, L., Petitti, A., Marani, R., & Milella, A. (2020). Internet of Robotic Things in Smart Domains: Applications and Challenges. *Sensors*, *20*(12), 3355.

Tiruvayipati, S. (2024). Methodology for Developing an IoT-based Parking Space Counter System using XNO. *Scalable Computing: Practice and Experience*, *25*(2), 800-811.

Tsang, Y., Wu, C., Ip, W., & Shiau, W.-L. (2021). Exploring the intellectual cores of the blockchain–Internet of Things (BIoT). *Journal of Enterprise Information Management*, *34*(5), 1287-1317.

Wei, T., Feng, W., Chen, Y., Wang, C.-X., Ge, N., & Lu, J. (2021). Hybrid Satellite-Terrestrial Communication Networks for the Maritime Internet of Things: Key Technologies, Opportunities, and Challenges. *IEEE Internet of Things Journal*, *8*(11), 8910-8934.