Department of I & CT
MIT, Manipal

Collision Detection

ICT 3143 Embedded Systems Lab Mini-Project

Vth Sem B.Tech (CCE)

230953022 –Archit Srivastava

230953034 –Maaheer Furia

230953038 – Sridha Rajendra

November, 2025

ABSTRACT

The Collision Detection System detects objects in real-time and notifies the user if the object is within a dangerous distance to avoid a collision. Using the LPC1768 ARM Cortex-M3 microcontroller and the HC-SR04 ultrasonic sensor, the system works by measuring the time taken by the ultrasonic waves to reflect back from nearby objects to detect the obstacle accurately. According to the Echo-based distance calculation, the distance of any entity from and wall is determined by the system.

The system indicates using a Red LED (lighting ON) whenever the distance is short (< 20 cm) and a blinking buzzer is activated in case the distance is critically short (<10 cm). Along with LED indications, a 16×2 LCD is used to continuously display the distance in centimetres. One will be able to access distance accurately, ensuring limited issues of crashing. Vehicle safety systems, robot navigation, and smart occupancy detection in parking can all benefit from the technology behind the system.

This project demonstrates how embedded systems can be harnessed to address real-world problems. It does this through effective sensing, feedback and human-centred design.

Keywords: Collision Detection, LPC1768, Ultrasonic Sensor, Embedded System, LCD Display, Blinking LED, Parking Assistance.

CONTENTS

List of Figures

1. Introduction

The Collision Detection System uses sensors that deploy ultrasonic technology to create a system of distance detection system. This project uses the HC-SR04 Ultrasonic Distance sensor integrated with the ARM Cortex-M3 LPC1768 microcontroller to detect objects and measure distance in real time with an accuracy of up to 3 mm. The chosen sensor for the prototype was the HC-SR04. The device uses ultrasonic waves at 40 kHz and reads distances from 2 centimetres to 4 meters, and can be used in many situations.

The LPC1768 microcontroller works with the HC-SR04 sensor, where the ultrasonic pulses are transmitted and the time taken for the echoes to be received is measured. The speed of sound in air is nearly 343 meters per second, and the distance is calculated based on this value. The HC-SR04 Sensor, which measures distance and detects objects, can be constructed using essential components. It has a transmitter which emits ultrasonic waves, a receiver which gets reflected echoes and a control circuit which controls timing and noise filtering. The sensor has four essential pins: 1) VCC (which is the power), 2) TRIG (which triggers pulses), 3) ECHO (which receives echoes), and 4) GND (which is the ground).

The sensor uses sound waves that bounce back from an object to observe its distance. The system measures the distance by measuring the time difference between emission and reception. The HC-SR04 needs a precise trigger pulse to start sending out 40 kHz ultrasounds. Ultrasonic sensing is used in high-precision and versatile applications in robotics, automation, and industrial applications in various fields..

2. Methodology

a. Components Required

- LPC1768 Microcontroller:

The ARM Cortex-M3-based LPC1768 microcontroller acts as the central controller, managing all aspects of the collision detection system—sensor activation, lighting up LEDs, and triggering the buzzer. It efficiently processes data from connected sensors and controls outputs to ensure fast and reliable operation.

- HC-SR04 Ultrasonic Sensor:

This sensor is responsible for detecting the presence and distance of objects. It sends distance data to the microcontroller for analysis.

- LED:

Eight LEDs (on P0.4–P0.11) turn on progressively as objects approach, and an additional LED on P0.17 blinks to indicate a critically short distance (< 20 cm).

- Buzzer:

Activated by the microcontroller upon detecting an object in close proximity, emitting a loud sound to alert the user of an imminent collision event..

- LCD Display (16×2):

Provides real-time feedback by displaying the measured distance and indicating the object distance to support collision detection. It displays only the measured distance in centimetres (with two decimal places) on the LCD, without predefined status messages.

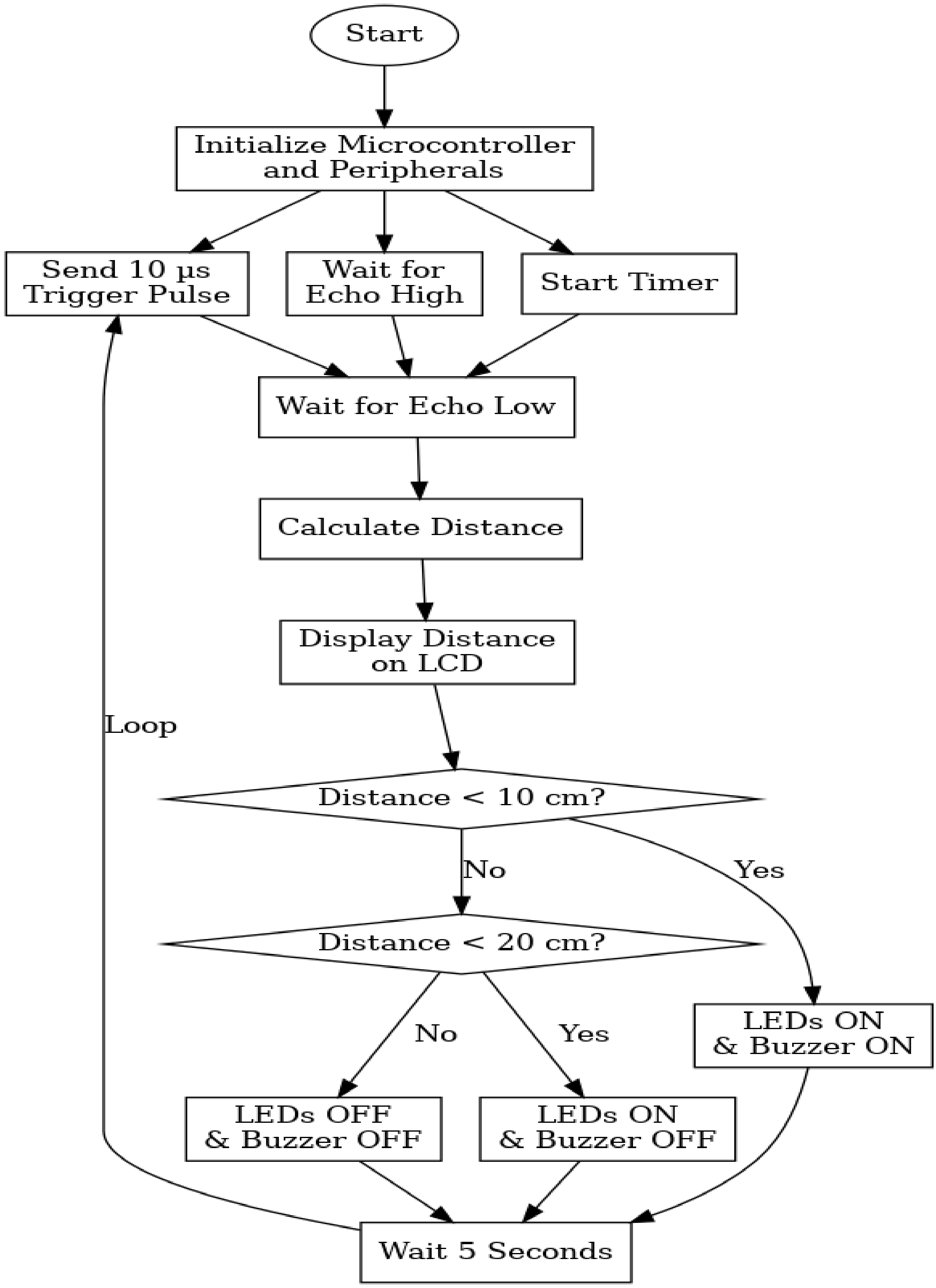
- Connecting Cables (FRC, Data, Jumper):

Used to connect the components to the LPC1768, ensuring stable communication and power flow across the system.

b. Flow Diagram

- Power Supply: Provides regulated power (typically 5V) to the LPC1768 and all connected components.
- LPC1768 Microcontroller: Central control unit interfacing with inputs/outputs and processing distance data.
- Input Block – Ultrasonic Sensor (HC-SR04): Trigger on P0.15 and Echo on P0.16 to measure object distance.
- Output Blocks – LCD Display: Shows real-time distance and status messages.

- Output Blocks – LEDs (P0.4 to P0.11): Visual distance indicator, LEDs turn on progressively as objects approach.
- Output Blocks – Buzzer: Emits a warning sound within a critical distance range.



c. Connection Description

- LCD Display:

The 16×2 LCD is connected to the LPC1768 for real-time visual feedback. RS → P0.27, E → P0.28. Data lines D4–D7 → P0.23–P0.26. The display and backlight are powered at 5V with a common ground.

- Ultrasonic Sensor:

Trigger (T) → P0.15 sends ultrasonic pulses; Echo (R) → P0.16 receives the returning pulse. Distance is computed from the pulse width.

- LEDs:

Eight LEDs are connected as outputs with series resistors (220–330 Ω): eight on P0.4–P0.11 via CNA1, plus an additional LED on P0.17.

- Buzzer:

Negative terminal → P2.0 (microcontroller control), positive terminal → 5V. When P2.0 is asserted, the buzzer emits an alert.

d. Method

(i) System start-up and safety defaults

- Configure GPIO directions for all devices.
  - **Ultrasonic:** TRIG = P0.15, ECHO = P0.16
  - **LEDs (8-bar + extra):** P0.4–P0.11 and an additional LED at P0.17.
  - **LCD (4-bit):** D4–D7 → P0.23–P0.26, RS → P0.27, E → P0.28
  - **Buzzer:** P0.22 (use driver transistor if required)
- Initialize TIMER0 for microsecond timing (used for the 10 µs trigger pulse and echo timing).
- Apply safe defaults: LEDs OFF, buzzer OFF, LCD cleared.
- Define a global **echo timeout ≈ 30 ms** to avoid blocking when there's no return.

(ii) Measurement cycle: Trigger → Echo → Distance

- Send a **10 µs** HIGH pulse on **TRIG (P0.15)** to start an ultrasonic ping.

- Wait for **ECHO (P0.16)** to go HIGH (up to the timeout).
    - If ECHO never rises → mark this reading **INVALID**.
- When ECHO is HIGH, measure how long it stays HIGH (µs).
- Convert time to one-way distance:
- distance_cm = (echo_us × 0.00343) / 2
- Clamp distance to a practical range **[2 … 400] cm**. If the reading was INVALID, set the distance to a safe placeholder (e.g., **200 cm**) so the UI remains stable.

(iii) Cleaning and smoothing the readings

- Maintain a **3-sample moving average** of distance to reduce jitter from soft or angled surfaces.
- **Outlier reject:** if a new sample differs by **>≈15 %** from the current mean, ignore it once and use the next cycle's value.

(iv) Zone classification with hysteresis

We convert the smoothed distance into clear action zones:

- **SAFE:** > **50 cm**
- **CAUTION: 20–50 cm**
- **CRITICAL:** < **20 cm**

To prevent flicker at boundaries, use **±2 cm hysteresis** at 20 cm and 50 cm.
Example: Once we enter CAUTION from SAFE, we only return to SAFE after crossing more than **52 cm**.

(v) User feedback logic (LEDs, buzzer, LCD)

- **LEDs (P0.4–P0.11):** show a proximity bar that **fills as distance decreases** (scaled for 0–80 cm).
- **Buzzer (P0.22):**
    - SAFE → **OFF**
    - CAUTION → **short beeps (~2 Hz)**
    - CRITICAL → **continuous tone**
- **LCD (16×2):** Display the measured distance in centimetres (with two decimal places) on the LCD, without fixed status messages.

(vi) Loop cadence and responsiveness

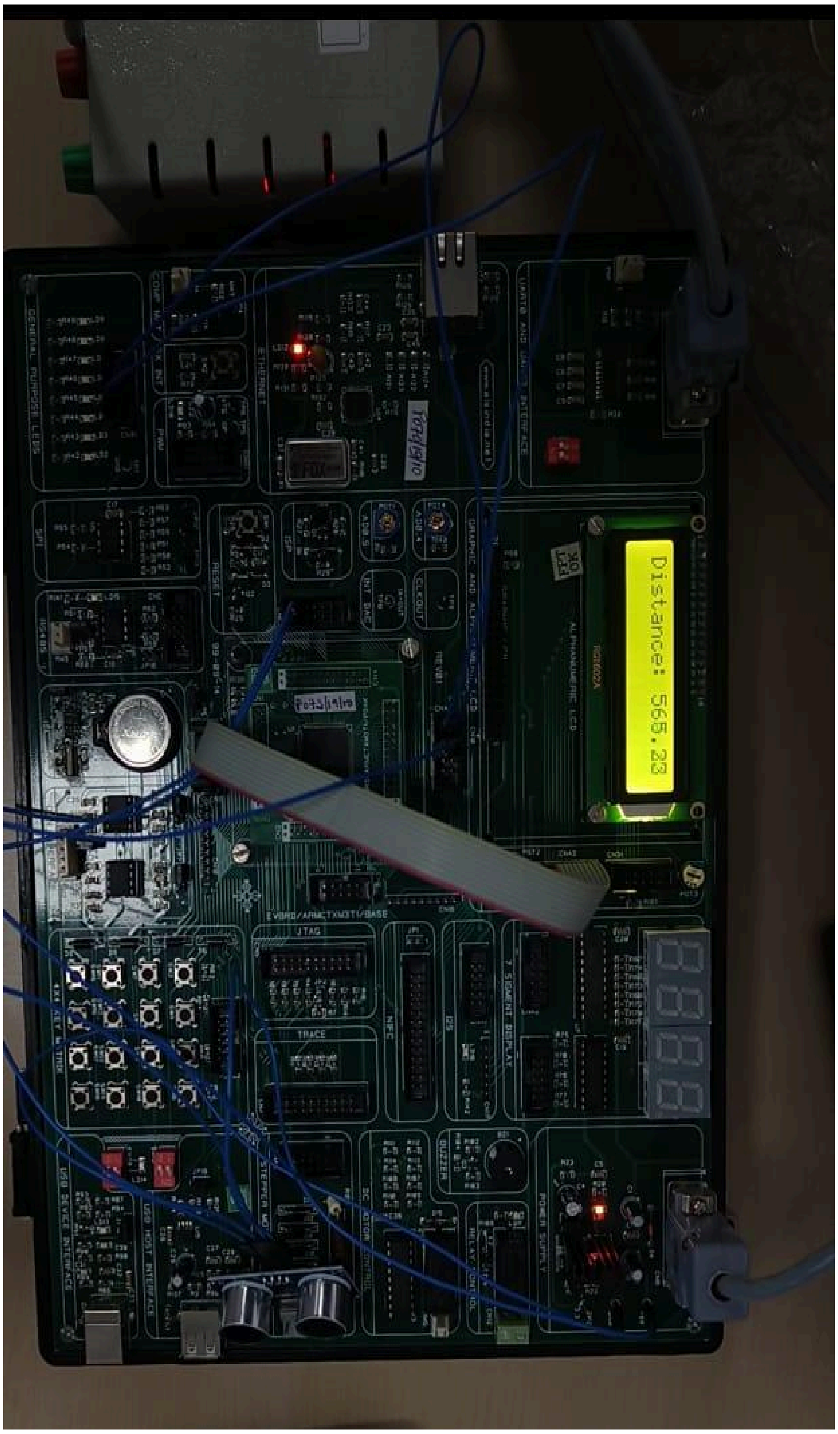
- Run the full measurement-to-UI loop at **~12 Hz** (every **≈80–90 ms**).
  This rate feels responsive for human reaction times, avoids LED flicker, and keeps the buzzer patterns distinct.

(vii) Optional calibration hook

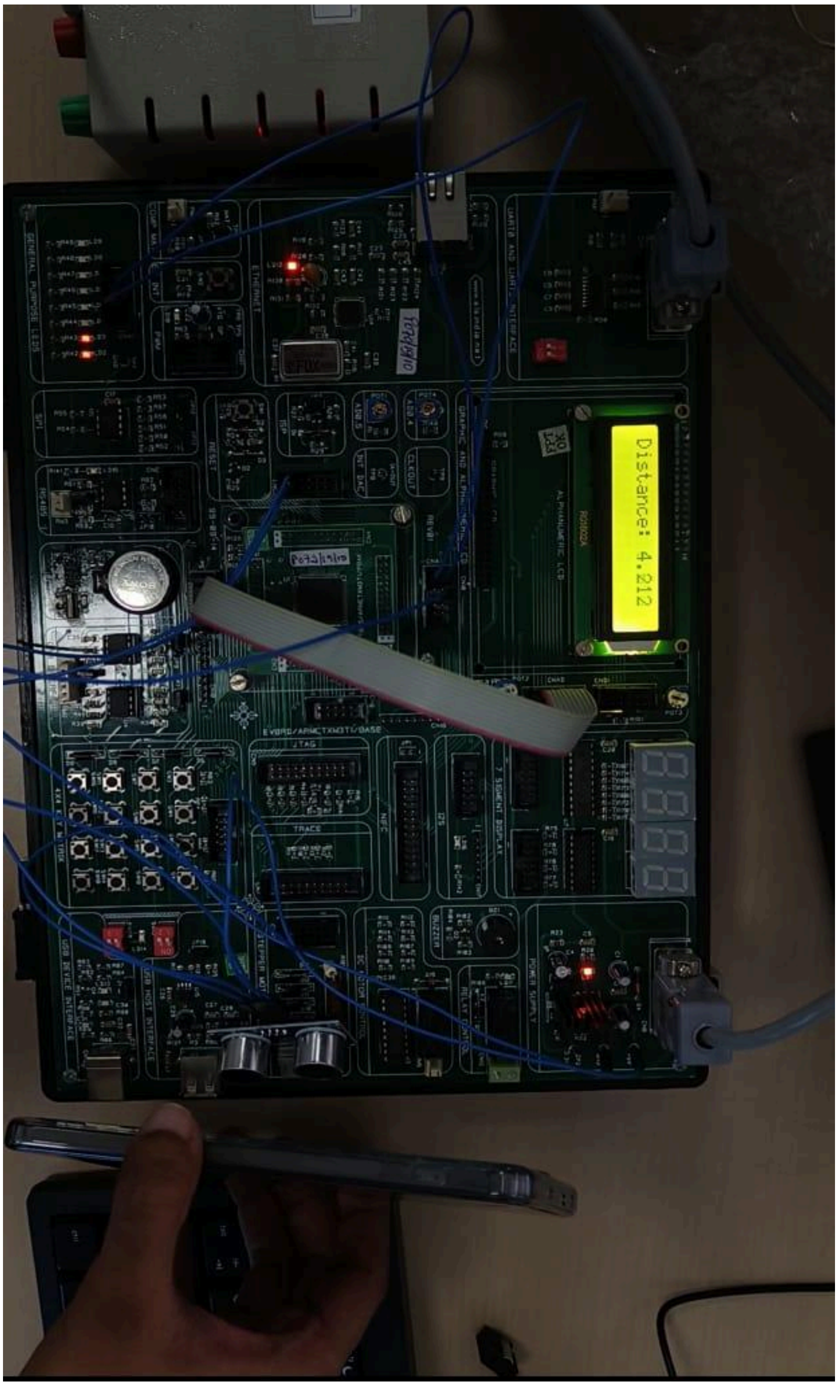
- Add a service mode to average ~**32 readings** at a known reference distance and compute a small offset (temperature / mounting angle).
- Store and apply the offset at boot for slightly tighter accuracy.

3. Results and Discussion

a. Photos of the result



Real-time distance calculation



LED Blinking

b. Working and Relevance of the System

Ultrasonic sensors play a crucial role in embedded safety systems due to their ability to detect nearby objects and measure distance accurately in real time. In this Collision Detection System, the HC-SR04 ultrasonic sensor is used to continuously monitor the distance between the system and surrounding obstacles. When an object comes too close—within a predefined threshold—the system triggers both visual (LEDs) and auditory (buzzer) alerts, allowing the user or system to respond before an impact occurs.

This type of real-time proximity-based collision detection is particularly useful in constrained environments such as robotics, warehouse automation, or low-speed vehicle navigation, where quick feedback is essential to prevent contact. The system works by emitting ultrasonic pulses and calculating the time it takes for the echo to return, determining the object's distance. If the measured distance drops below the critical limit, the system activates warnings to signal a potential collision.

Compared to more complex automotive applications like automatic braking or blind spot detection, this implementation focuses on providing basic, reliable alerts in response to immediate proximity threats. It serves as a practical demonstration of how ultrasonic sensors can be applied in embedded systems for low-cost, real-time collision detection.

4. References

1. LPC1768 Microcontroller Documentation:
   https://www.nxp.com/docs/en/user-guide/UM10360.pdf
2. Keil uVision IDE: https://www.keil.com/mdk5/
3. Ultrasonic Sensor (HC-SR04) Working Principles:
   https://techatronic.com/hc-sr04-ultrasonic-sensor-working-and-explanation/
4. LCD Display Interfacing with LPC1768:
   https://www.bascom-tutorials.com/lcd-interfacing-with-lpc176

5. C Code with Comments

```
/************************************************************
 * Project      : Ultrasonic Distance Measurement System
 * Microcontroller : NXP LPC1768 (ARM Cortex-M3)
 * Peripherals  : HC-SR04 Ultrasonic Sensor, 16x2 LCD (4-bit mode),
 *              LEDs on P0.4-P0.11, Buzzer on P0.22 *
 * Description  :
 *   - Measures distance using ultrasonic sensor (TRIG P0.15, ECHO P0.16)
 *   - Displays distance on 16x2 LCD
 *   - Turns ON LEDs & buzzer if object < 7 cm
 ************************************************************/


#include <stdio.h>
#include <LPC17xx.h>
#include <string.h>


// ---------------- PIN DEFINITIONS ----------------
#define LED_Pinsel     0xFF        // LEDs on P0.4 - P0.11
#define TRIGGER_PIN    (1 << 15)   // Ultrasonic Trigger Pin (Output)
#define ECHO_PIN       (1 << 16)   // Ultrasonic Echo Pin (Input)


// ---------------- GLOBAL VARIABLES ----------------
char ans[20] = "";       // LCD string buffer
int temp, temp1, temp2 = 0;
int flag = 0, flag_command = 0;
int i, j, k, l, r;
int echoTime = 5000;     // Time duration of echo pulse
float distance = 0;      // Calculated distance in cm


// ---------------- FUNCTION DECLARATIONS ----------------
void lcd_wr(void);
void port_wr(void);
void delay(int r1);
void timer_start(void);
float timer_stop(void);
void timer_init(void);
void dealy_in_US(unsigned int microseconds);
void dealy_in_MS(unsigned int milliseconds);


// ------------------------------------------------------
// MICROSECOND DELAY USING TIMER0
// ------------------------------------------------------
void dealy_in_US(unsigned int microseconds){
    LPC_TIM0->TCR = 0x02;            // Reset Timer
    LPC_TIM0->PR = 0;                // Prescaler = 0
    LPC_TIM0->MR0 = microseconds - 1; // Match value
    LPC_TIM0->MCR = 0x01;            // Interrupt on match
    LPC_TIM0->TCR = 0x01;            // Enable timer


    while ((LPC_TIM0->IR & 0x01) == 0); // Wait for match flag


    LPC_TIM0->TCR = 0x00;            // Stop timer
    LPC_TIM0->IR = 0x01;             // Clear interrupt flag
}
```

```
void dealy_in_MS(unsigned int milliseconds){
    dealy_in_US(milliseconds * 1000);
}


// ----------------------------------------------------------
// TIMER INITIALIZATION FOR ECHO MEASUREMENT
// ----------------------------------------------------------
void timer_init(void){
    LPC_TIM0->CTCR = 0x0;          // Timer mode
    LPC_TIM0->PR = 11999999;        // 12 MHz clock division
    LPC_TIM0->TCR = 0x02;          // Reset timer
}


void timer_start(void){
    LPC_TIM0->TCR = 0x02;          // Reset timer
    LPC_TIM0->TCR = 0x01;          // Start timer
}


float timer_stop(void){
    LPC_TIM0->TCR = 0x00;          // Stop timer
    return LPC_TIM0->TC;          // Return captured value
}


// ----------------------------------------------------------
// LEVEL DELAY FUNCTION (SOFTWARE)
// ----------------------------------------------------------
void delay(int r1){
    for (r = 0; r < r1; r++);
}


// ----------------------------------------------------------
// LCD LOW-LEVEL WRITE (4-BIT INTERFACE)
// ----------------------------------------------------------
void port_wr(void){
    LPC_GPIO0->FIOPIN = temp2 << 23;   // Data on P0.23-P0.26


    if (flag_command == 0)
        LPC_GPIO0->FIOCLR = 1 << 27;   // RS = 0 → command
    else
        LPC_GPIO0->FIOSET = 1 << 27;   // RS = 1 → data


    LPC_GPIO0->FIOSET = 1 << 28;       // EN = 1 (Latch)
    for (j = 0; j < 50; j++);
    LPC_GPIO0->FIOCLR = 1 << 28;       // EN = 0
    for (j = 0; j < 10000; j++);
}


void lcd_wr(void){
    temp2 = (temp1 >> 4) & 0xF;  // Send upper nibble
    port_wr();
    temp2 = temp1 & 0xF;        // Send lower nibble
    port_wr();
}


// ----------------------------------------------------------
```

```
// MAIN PROGRAM
// -------------------------------------------------------
int main(){
   int command_init[] = {3,3,3,2,2,0x01,0x06,0x0C,0x80};  // LCD init sequence


   SystemInit();
   SystemCoreClockUpdate();
   timer_init();


   // GPIO CONFIGURATION
   LPC_PINCON->PINSEL0 &= 0xfffff00f;    // LEDs P0.4-P0.11
   LPC_PINCON->PINSEL0 &= 0x3fffffff;    // TRIGGER P0.15 GPIO
   LPC_PINCON->PINSEL1 &= 0xfffffff0;    // ECHO P0.16 GPIO


   LPC_GPIO0->FIODIR |= TRIGGER_PIN | (1<<22);    // Trigger + Buzzer output
   LPC_GPIO1->FIODIR |= 0 << 16;                  // Echo input
   LPC_GPIO0->FIODIR |= LED_Pinsel << 4;          // LEDs output
   LPC_GPIO0->FIODIR |= 0xF << 23 | 1<<27 | 1<<28;// LCD pins


   // LCD INITIALIZATION
   flag_command = 0;
   for (i = 0; i < 9; i++){
      temp1 = command_init[i];
      lcd_wr();
      for (j = 0; j < 30000; j++);
   }


   while (1){
      // SEND 10us TRIGGER PULSE
      LPC_GPIO0->FIOSET |= TRIGGER_PIN;
      dealy_in_US(10);
      LPC_GPIO0->FIOCLR |= TRIGGER_PIN;


      // WAIT FOR ECHO HIGH
      while (!(LPC_GPIO0->FIOPIN & ECHO_PIN));


      timer_start();   // START TIMER


      // WAIT UNTIL ECHO GOES LOW
      while (LPC_GPIO0->FIOPIN & ECHO_PIN);


      echoTime = timer_stop();    // READ TIMER VALUE


      distance = (0.00343 * echoTime) / 2;  // Convert to cm


      sprintf(ans, " Distance: %.3f", distance);


      // CLEAR LCD AND DISPLAY RESULT
      flag_command = 0;
      temp1 = 0x01;
      lcd_wr();
```

```
        flag_command = 1;
        for (i = 0; ans[i] != '\0'; i++){
           temp1 = ans[i];
           lcd_wr();
           for (j = 0; j < 300; j++);
        }
       dealy_in_MS(100);

 // ALERT IF DISTANCE TOO SMALL
             if(distance <20)
           {
                    LPC_GPIO0->FIOSET = LED_Pinsel << 4;  // LEDs ON
                    if (distance < 10)
                    {
                            LPC_GPIO0->FIOSET = 1 << 22;          // Buzzer ON
                    }
                    else
                    {
                            LPC_GPIO0->FIOCLR = 1 << 22;          // Buzzer OFF
                    }

           }
           else {
                    LPC_GPIO0->FIOCLR = LED_Pinsel << 4;  // LEDs OFF
                    LPC_GPIO0->FIOCLR = 1 << 22;          // Buzzer OFF
        }

        dealy_in_MS(5000);
    }
}
```

Similarity Index Report

/25, 4:24 PM Turnitin - Originality Report - CollisionDetectionProjectReport.pdf

preferences

turnitin
Originality Report

Processed on: 08-Nov-2025 10:01 AM +08
ID: 2807256649
Word Count: 2319
Submitted: 1

CollisionDetection
By Archit Srivastava

Similarity Index	Similarity by Source	
1%	Internet Sources:	0%
	Publications:	1%
	Student Papers:	N/A

Document Viewer

nclude quoted | include bibliography | excluding matches < 3 words

mode: show highest matches together

CT MIT, Manipal Collision Detection ICT 3143 Embedded Systems Lab Mini-Project Vth Sem B.Tech (CCE) 230953022 –Archit Srivastava 230953034 –Maaheer Furia 230953038 – Sridha Rajendra November, 2025 ABSTRACT The Collision Detection System detects objects in real-time and notifies the user if the object is within a dangerous distance to avoid a collision. Using the LPC1768 ARM Cortex-M3

microcontroller and the HC-SR04 ultrasonic sensor, the 2

10 cm). Along with LED indications, a 16×2 LCD is used to continuously display the distance in centimetres. One will be able to access distance accurately, ensuring limited issues of crashing. Vehicle safety systems, robot navigation, and smart occupancy detection in parking can all benefit from the technology behind the system. This project demonstrates how embedded systems can be harnessed to address real-world problems. It does this through effective sensing, feedback and human-centred design. Keywords: Collision Detection, LPC1768, Ultrasonic Sensor, Embedded System, LCD Display, Blinking LED, Parking Assistance. CONTENTS 1. Introduction 2. Methodology a. Components Required b. Flow Diagram c. Connection Description d. Method 3. Results and Discussion a. Photographs b. Working and Relevance of the System 4. References 5. C Code with Comments List of Figures 1. Flow Diagram 2. Connection Diagram 3. Photographs of the System 1. Introduction The Collision Detection System uses sensors that deploy ultrasonic technology to create a system of distance detection system. This project uses the HC-SR04 Ultrasonic Distance sensor integrated with the ARM Cortex-M3 LPC1768 microcontroller to detect objects and measure distance in real time with an accuracy of up to 3 mm. The chosen sensor for the prototype was the HC-SR04. The device uses ultrasonic waves at 40 kHz and reads distances from 2 centimetres to 4 meters, and can be used in many situations. The LPC1768 microcontroller works with the HC-SR04 sensor, where the ultrasonic pulses are transmitted and

the time taken for the echoes **to be received is** measured. **The** 1

22; // Buzzer OFF } } dealy_in_MS(5000); Similarity Index Report

1 — 1% match (Singireddy, Kalyan Srinivas Reddy. "Active Safety System Model to Avoid Rear-End Vehicle Collision Using Controlled Area Network (CAN).", Texas A&M University - Kingsville, 2020)
Singireddy, Kalyan Srinivas Reddy. "Active Safety System Model to Avoid Rear-End Vehicle Collision Using Controlled Area Network (CAN).", Texas A&M University - Kingsville, 2020

2 — 1% match (Holliday, Tyler. "Modeling and Prototyping a Modular, Low-Cost Collision Avoidance System for UAVs.", Montana Tech of The University of Montana, 2020)
Holliday, Tyler. "Modeling and Prototyping a Modular, Low-Cost Collision Avoidance System for UAVs.", Montana Tech of The University of Montana, 2020