**SAFEPARK DISTANCE GAURD**

**ICT 2242 Embedded Systems lab miniproject**

**Reg. No Your Name(s)**

**220911536 Prashast Saxena**

**220911534 Arnav Goel**

**220911532 Vikranth Reddy**

**220311524 Kiran Kishore**

**April 2024**

**Contents:**

1. Abstract
2. Introduction
3. List of Figures
4. Methodology
   1. Components required
   2. Block Diagram
   3. Description about the connection
   4. Method
5. Results and Discussion
6. References
7. C code with comments
8. Similarity Index report

**Abstract:**

The "SafePark Distance Guard" emerges as a groundbreaking solution for enhancing parking efficiency in bustling parking environments. Employing cutting-edge ultrasonic sensor technology, this device precisely calculates the space between parked vehicles in realtime. The collected parking distance information is promptly displayed on an LCD screen, providing users with instant feedback. An intelligent warning mechanism, featuring an LED indicator, triggers when the measured distance between vehicles falls below the recommended 3 feet, promoting safe parking practices. With its capability for continuous monitoring, this system is well-suited for various settings, including commercial parking areas, public garages, and residential parking spots. By optimizing parking space utilization and ensuring vehicles are parked safely apart, the "SafePark Distance Guard" not only enhances convenience but also reduces the likelihood of minor vehicle collisions, thus elevating the overall parking experience.

**Introduction:**

a) Scope: The project's goal is to create a parking distance monitoring system utilizing an ultrasonic sensor. This system will display real-time parking distance information on an LCD screen and feature a built-in LED indicator to signal when distances are less than 3 feet, promoting safe parking and public safety in busy parking areas.

b) Project Overview: The " SafePark Distance Guard " is an innovative system designed to optimize parking in crowded areas. It utilizes advanced ultrasonic sensor technology to measure the distance between parked vehicles in real-time accurately. The collected data is immediately displayed on an LCD screen, providing users with instant insights. An intelligent alert system with an LED indicator activates when the distance between vehicles falls below 3 feet, encouraging safe parking practices. This system is suitable for various applications, including commercial parking lots, public garages, and residential parking spaces. By ensuring vehicles are parked at a secure distance from each other, it improves both convenience and safety, reducing the risk of minor vehicle damage and enhancing the overall parking experience.

c) Problem Statement: Develop a " SafePark Distance Guard " using ultrasonic sensors to provide real-time parking distance information on an LCD screen. Include an LED alert for distances below 3 feet to promote safe parking and prevent vehicle damage in congested areas.

d) Objective: The primary objective of the " SafePark Distance Guard " project is to create a reliable and user-friendly parking distance monitoring system. This system utilizes an ultrasonic sensor to measure the distance between parked vehicles in real-time precisely. The collected data is promptly displayed on the ARM microcontroller kit's LCD screen, offering users immediate feedback. Additionally, an intelligent alert system featuring an LED indicator activates when the measured distance falls below the recommended safety threshold of 3 feet. By combining advanced technology with user safety, the ParkSafe Proximity Monitor aims to promote secure parking and enhance the convenience and safety of parking in crowded environments.

**List of Figures:**

1. Fig 1: Ultrasonic HC-SR04 module timing diagram.
2. Fig 2: Circuit Diagram: LPC1768 interfaced with HC-SR04 ultrasonic sensor, LEDs, LCD, and buzzer, displaying the respective pin configurations.
3. Fig 3: Distance greater than 20cm.
4. Fig 4: Distance less than 20cm.

**Methodology:**

1. Components Required

• LPC 1758 microcontroller

• Power supply (+5V)

• Cross cable for programming and serial communication

• One working USB port in the host computer system and PC

• HC-SR04 Ultrasonic Distance Sensor

• USB to B type cable

Software Requirements:

• Language: C

• IDE : Keil MicroVision

• Application: Flash Magic

1. Block Diagram

A diagram of a computer

Description automatically generated

Fig 1: Circuit Diagram: LPC1768 interfaced with HC-SR04 ultrasonic sensor, LEDs, LCD, and buzzer, displaying the respective pin configurations.

1. Description of the connection

In terms of pinout, the module features four pins: VCC (+5V), TRIG, ECHO, and GND.

1. The trigger pin is given a short pulse of 10us.

2. Upon receiving a trigger pulse, the HC-SR04 Module emits a burst of eight ultrasonic pulses at 40 kHz.

3. Then, it outputs a HIGH for the time the sound waves take to reach back.

4. The duration of the high pulse is measured and subsequently utilized to determine the distance.

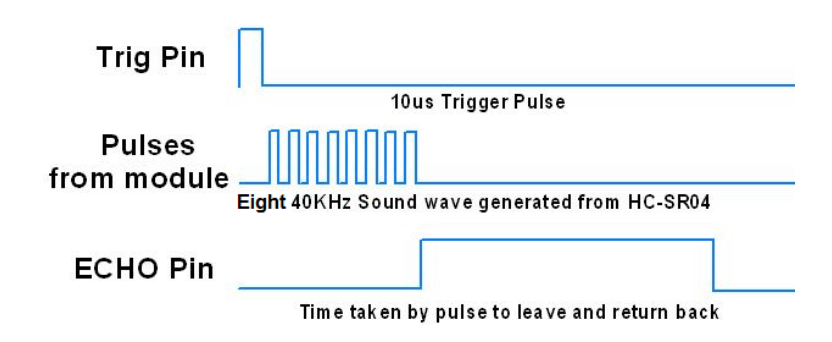


Fig 1: Ultrasonic HC-SR04 module timing diagram.

1. Method

The HC-SR04 Ultrasonic Distance/Ranging Sensor operates by emitting ultrasound to determine the distance to an object. Ultrasound, which operates beyond the audible range (> 20 kHz), serves as the foundation for its functionality. With a detection range from 2 centimeters to 4 meters, this module utilizes a 40 kHz ultrasound signal to gauge the distance between itself and any object within its detection zone.

Pin Configuration: The module comprises four pins: VCC (+5V), TRIG, ECHO, and GND.

Like SONAR technology, this ultrasonic sensor consists of two transducers—one for emitting ultrasound and the other for receiving the echo. Distance computation relies on the speed of sound in the air, which is approximately 343 m/s.

**Results and Discussion**:

A close-up of a circuit board

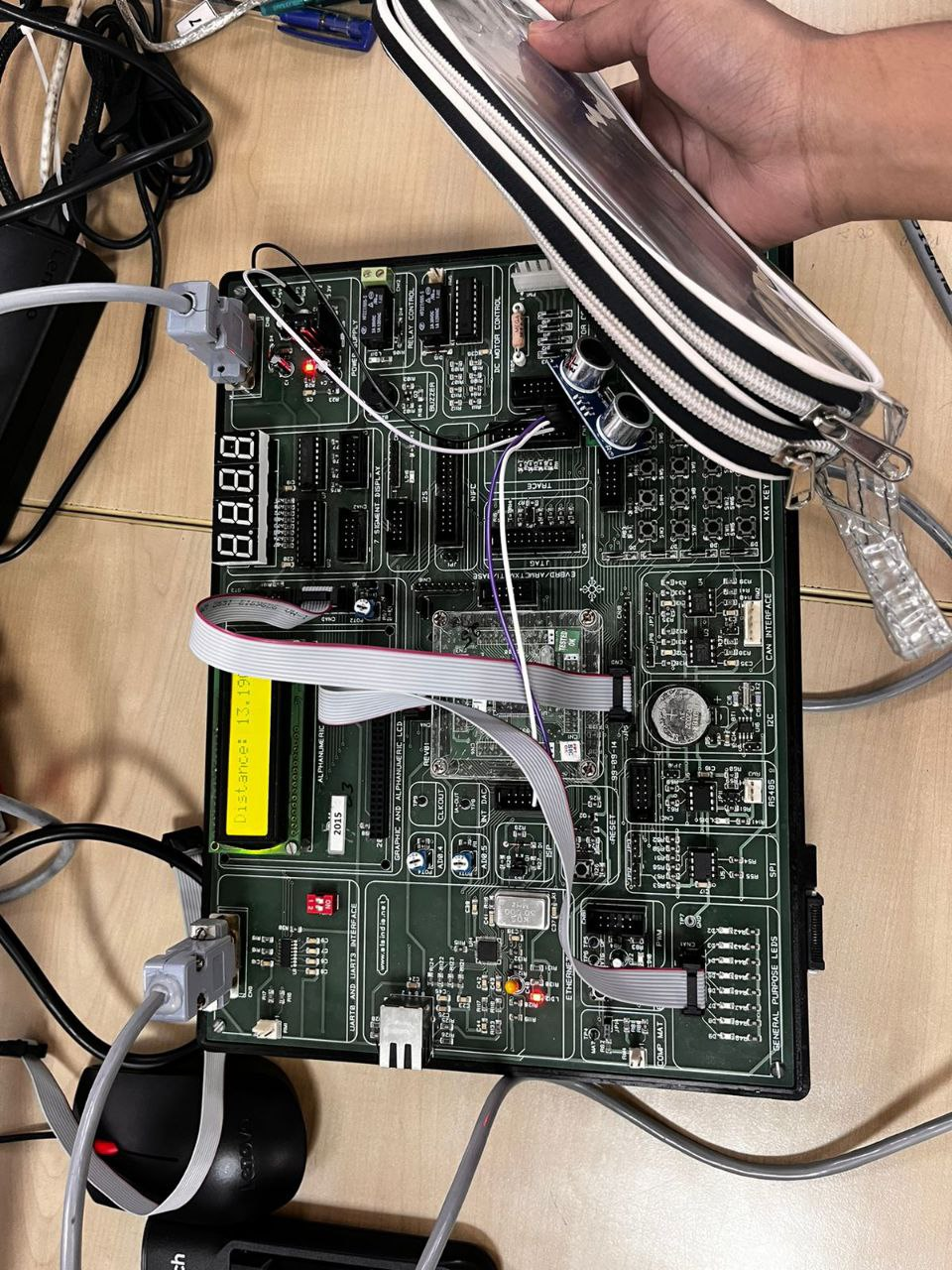
Description automatically generated

Fig 3: Distance greater than 20cm Fig 4: Distance less than 20cm

When the ultrasonic sensor detects a distance > 20 cm, the system measures the distance between the sensor and the object and displays it on the LCD screen, as depicted in Figure 3. Conversely, when the detected distance is < 20 cm, the system promptly measures and displays it on the LCD screen. Additionally, as an alert mechanism, the system initiates a continuous beeping sound from the buzzer as long as the object remains within this proximity.

**References:**

[1] Robocraze. (n.d.). What is Ultrasonic Sensor: Working Principle & Applications. No Author info. Retrieved from [https://robocraze.com/blogs/post/what-is-ultrasonic-sensor - header-15](https://robocraze.com/blogs/post/what-is-ultrasonic-sensor%20-%20header-15)

[2] Aliew, F. (2022). An Approach for Precise Distance Measuring Using Ultrasonic Sensors. Eng. Proc., 24(1), 8. Retrieved from <https://doi.org/10.3390/IECMA2022-12901>

[3] Sahoo, Biren & Shahjad, & Tanwar, Prakash. (2021). Real-Time Smart Parking: Challenges and Solution using Machine learning and IoT. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 451-458. 10.32628/CSEIT217295.Retrived from <https://doi.org/10.32628/CSEIT217295>

**C code with comments:**

#include <stdio.h>

#include <LPC17xx.h>

#include <string.h>

// Defining Constants

#define LEDPINS 0xff // P0.4-0.11 (LEDs)

#define TRIG (1 << 15) // P0.15 (Trigger Pin)

#define ECHO (1 << 16) // P0.16 (Echo Pin)

#define BUZZ (1<<13)

// Variable Declarations

char ans[20] = "";

int temp, temp1, temp2 = 0;

int flag = 0, flag\_command=0;

int i, j, k, l, r, echoTime = 5000;

float distance = 0;

//Function Declarations

void lcdwrite(void);

void portwrite(void);

void delay(int r1);

void timer\_start(void);

float timer\_stop();

void timerinitialisation(void);

void delaymicro(unsigned int micro);

void delaymilli(unsigned int milli);

//Functions

void delaymicro(unsigned int micros)

{

 LPC\_TIM0->TCR = 0x02;

 LPC\_TIM0->PR = 0; // Set prescaler to the value of 0

 LPC\_TIM0->MR0 = microseconds - 1; // Set match register for 10us

 LPC\_TIM0->MCR = 0x01; // Interrupt on match

 LPC\_TIM0->TCR = 0x01; // Enable timer

 while ((LPC\_TIM0->IR & 0x01) == 0); // Wait for interrupt flag

 LPC\_TIM0->TCR = 0x00; // Stop the timer

 LPC\_TIM0->IR = 0x01; // Clear the interrupt flag

}

void delaymilli(unsigned int milli) // Using Timer0

{

 delaymicro(milliseconds \* 1000);

}

void timerinitialisation(void)

{

 // Timer for distance

 LPC\_TIM0->CTCR = 0x0;

 LPC\_TIM0->PR = 11999999; // 12Mhz  for LPC 1768

 LPC\_TIM0->TCR = 0x02; // Reset Timer

}

void timer\_start(void)

{

 LPC\_TIM0->TCR = 0x02; // Reset Timer

 LPC\_TIM0->TCR = 0x01; // Enable timer

}

float timer\_stop()

{

LPC\_TIM0->TCR = 0x0;

 return LPC\_TIM0->TC;

}

void delay(int r1)

{

 for (r = 0; r < r1; r++);

}

void portwrite()

{

 int j;

 LPC\_GPIO0->FIOPIN = temp2 << 23;

 if (flag\_command == 0)

 {

 LPC\_GPIO0->FIOCLR = 1 << 27;

 }

 else

 {

 LPC\_GPIO0->FIOSET = 1 << 27;

 }

 LPC\_GPIO0->FIOSET = 1 << 28;

 for (j = 0; j < 50; j++);

 LPC\_GPIO0->FIOCLR = 1 << 28;

 for (j = 0; j < 10000; j++);

}

void lcdwrite()

{

 temp2 = (temp1 >> 4) & 0xF;

 portwrite();

 temp2 = temp1 & 0xF;

 portwrite();

}

// Main Program

int main()

{

 int ledflag = 0;

 int lcdinit[] = {3, 3, 3, 2, 2, 0x01, 0x06, 0x0C, 0x80};

 SystemInit();

 SystemCoreClockUpdate();

 timerinitialisation();

 LPC\_PINCON->PINSEL0 &= 0xfffff00f; // Interface LEDs P0.4-P0.11

 LPC\_PINCON->PINSEL0 &= 0x3fffffff; // Interface TRIG P0.15

 LPC\_PINCON->PINSEL1 &= 0xfffffff0; // Interface ECHO P0.16

LPC\_PINCON->PINSEL4  &= 0xfcffffff;//interface BUZZ

 LPC\_GPIO0->FIODIR |= TRIG ; // Direction for TRIGGER pin and BUZZ1

LPC\_GPIO2->FIODIR |=BUZZ;

 LPC\_GPIO1->FIODIR |= 0 << 16; // Direction for ECHO PIN

 LPC\_GPIO0->FIODIR |= LEDPINS << 4; // Direction for LED

 LPC\_PINCON->PINSEL1 |= 0;

 LPC\_GPIO0->FIODIR |= 0XF << 23 | 1 << 27 | 1 << 28; // Direction For LCDs

 flag\_command = 0;

 for (i = 0; i < 9; i++)

 {

 temp1 = lcdinit[i];

 lcdwrite();

 for (j = 0; j < 30000; j++);

 }

 i = 0;

 flag = 1;

 LPC\_GPIO0->FIOCLR |= TRIG;

 while (1)

 {

 LPC\_GPIO0->FIOSET = 0x00000800;

 //  10us HIGH to the TRIG pin

 LPC\_GPIO0->FIOMASK = 0xFFFF7FFF;

 LPC\_GPIO0->FIOPIN |= TRIG;

 delaymicro(10);

 LPC\_GPIO0->FIOCLR |= TRIG;

 LPC\_GPIO0->FIOMASK = 0x0;

 while (!(LPC\_GPIO0->FIOPIN & ECHO))

 {

 // Wait till ECHO PIN becomes high

 }

 timer\_start();

 while (LPC\_GPIO0->FIOPIN & ECHO); // Wait till ECHO PIN becomes low

 echoTime = timer\_stop(); // Time is stored betwenn low anf high of echo pin

 distance = (0.00343 \* echoTime) / 2; //Calculations of Distance in cm

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

 flag\_command = 1;

 i = 0;

 flag\_command = 0;

 temp1 = 0x01;

 lcdwrite();

 flag\_command = 1;

 while (ans[i] != '\0')

 {temp1 = ans[i];

 lcdwrite();

 for (j = 0; j < 150000; j++);

 i++;

 }

 if (distance < 20)

 {

 LPC\_GPIO0->FIOSET = LEDPINS << 4;

LPC\_GPIO2->FIOSET = BUZZ;

delay(90000);

 }

 else

 {

LPC\_GPIO2->FIOCLR = BUZZ;

 LPC\_GPIO0->FIOCLR = LEDPINS << 4;

 }

 delay(90000);

 }

}

**Append similarity index report**