Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

School of Electronics and Communication Engineering

Second Year B. Tech. (ECE)

Microcontrollers Course Code: ECE214A

Reverse Parking Buzzer

By

Aditya Gaidhani (PB07)
Shivani Kanyal (PB08)
Amushya Sinha(PB09)

NAME OF THE GUIDE
Prof. Shruti Danve

AKNOWLEDGEMENT

We would like to express my special thanks to my subject Professor Shruti Danve Ma'am and our HOS for the school of electronics and telecommunication Dr. Vinaya Gohokar Ma'am who gave us the golden opportunity to do this project on the topic '**Reverse Buzzer Sensor'**. It helped us in doing a lot of Research and we as a team came to know about a lot of things related to this topic.

Finally, we would say that this project was a team effort and we helped each other a lot in finalizing this project within the limited time frame and implementing it well. Although we all did face some difficulties, but we all came over it and learnt a lot even during the COVID-19 pandemic.

Abstract:

There is a rapid development in technology which influencing the human life in several aspects due to rapid development in different fields, but we still need to adopt that technology such that we can make human life more easier to live. We know that in automobiles sector many countries are doing great and launching new products (vehicles). Instead of having many advantages and features they may have some disadvantages too and one of them in vehicles is reverse car parking. To rectify this problem, we have developed a system which will help us to park our vehicles in reverse direction without collision with any obstacle in its vicinity. The aim of our project is to present such a design which can identify the obstacles comes in the range while parking the car in reverse direction.

INDEX

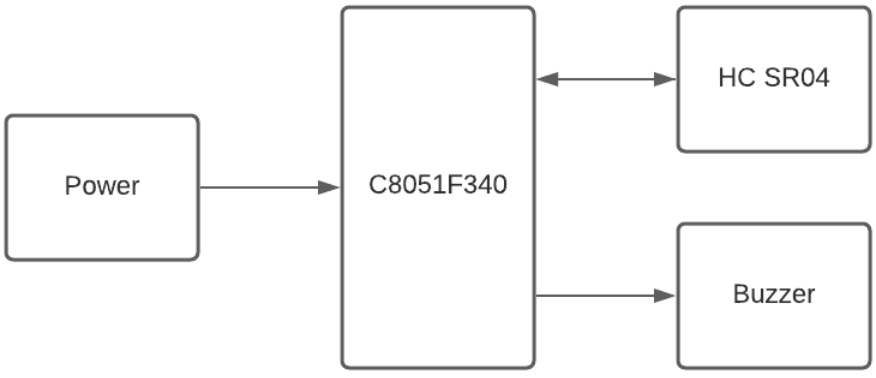
(1.) INTRODUCTION

As the population is increasing, the vehicles are also increasing and so the parking problem arises. There is a chance that a car may bump into other car while parking. Although the rear-view mirror gives a rough estimate of the distance with the obstacle behind, there is still a chance of collision. There is need to alert the driver when the car is about to collide with a nearby obstacle so that necessary precautions can be taken to avoid any damage.

Our system constitutes of a sensor which senses the distance of the nearby obstacle. This sensor when installed on the car alerts the driver when he is moving close to nearby obstacle and there is a chance of collision. The system can hence help avoid collision and damage, thereby helping save money.

(2.) HARDWARE DESIGN

(2.1) Block diagram



(2.2) Description

The ultrasonic sensor detects the distance between the car and object, and if the distance is less than 40cm then the sensor sends a signal to the microcontroller, then in turn the microcontroller sends a signal to the buzzer, and then it will beep with a predefined delay.

(2.3) Selection of Components and its specification

(2.3.1) C8051F340

C8051F340 device is a fully integrated mixed-signal System-on-a-Chip MCUs. Highlighted features are listed below:

- High-speed pipelined 8051-compatible microcontroller core
- Universal Serial Bus (USB) Function Controller with eight flexible endpoint pipes, integrated transceiver, and 1 kB FIFO RAM
- True 10-bit 200 ksps differential / single-ended ADC with analog multiplexer
- Precision internal calibrated 12 MHz internal oscillator and 4x clock multiplier
- Up to 64 kB of on-chip Flash memory
- Up to 4352 Bytes of on-chip RAM (256 + 4 kB)
- On-chip Power-On Reset, VDD Monitor, and Missing Clock Detector
- External Memory Interface (EMIF)
- SMBus/I2C, up to 2 UARTs, and Enhanced SPI serial interfaces implemented in hardware

(2.3.2) Ultrasonic sensor (HC-SR04)

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules include ultrasonic transmitters, receiver, and control circuit. The basic principle of work:

(1) Using IO trigger for at least 10us high level signal,

(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.

(3) IF the signal back, through high level, time of high output IO duration is the time from sending ultrasonic to returning.

Test distance = (high level time×velocity of sound (340 m/sec) / 2

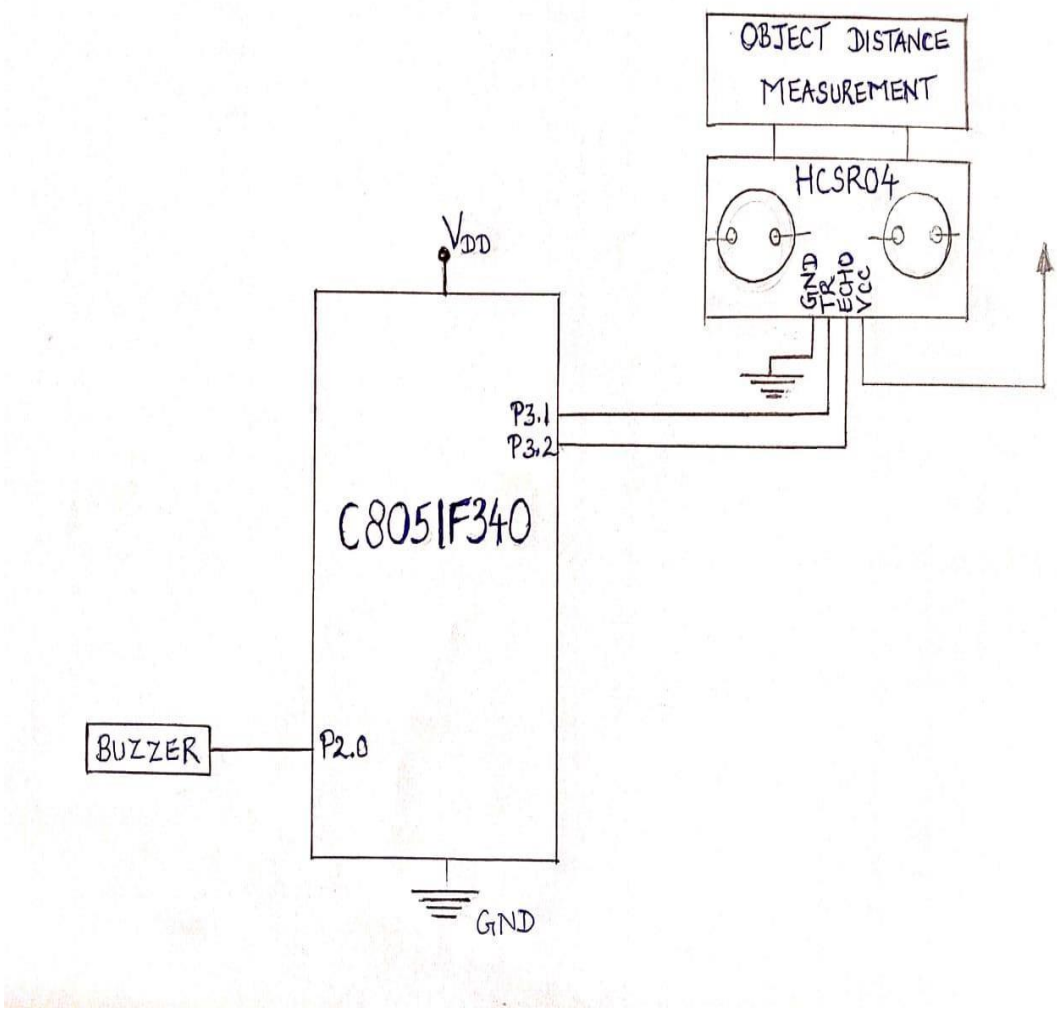
(2.3.3) Buzzer

Buzzer is a kind of voice device that converts audio model into sound signal It is mainly used to prompt or alarm.

According to different design and application, it can produce music sound flute sound, buzzer alarm

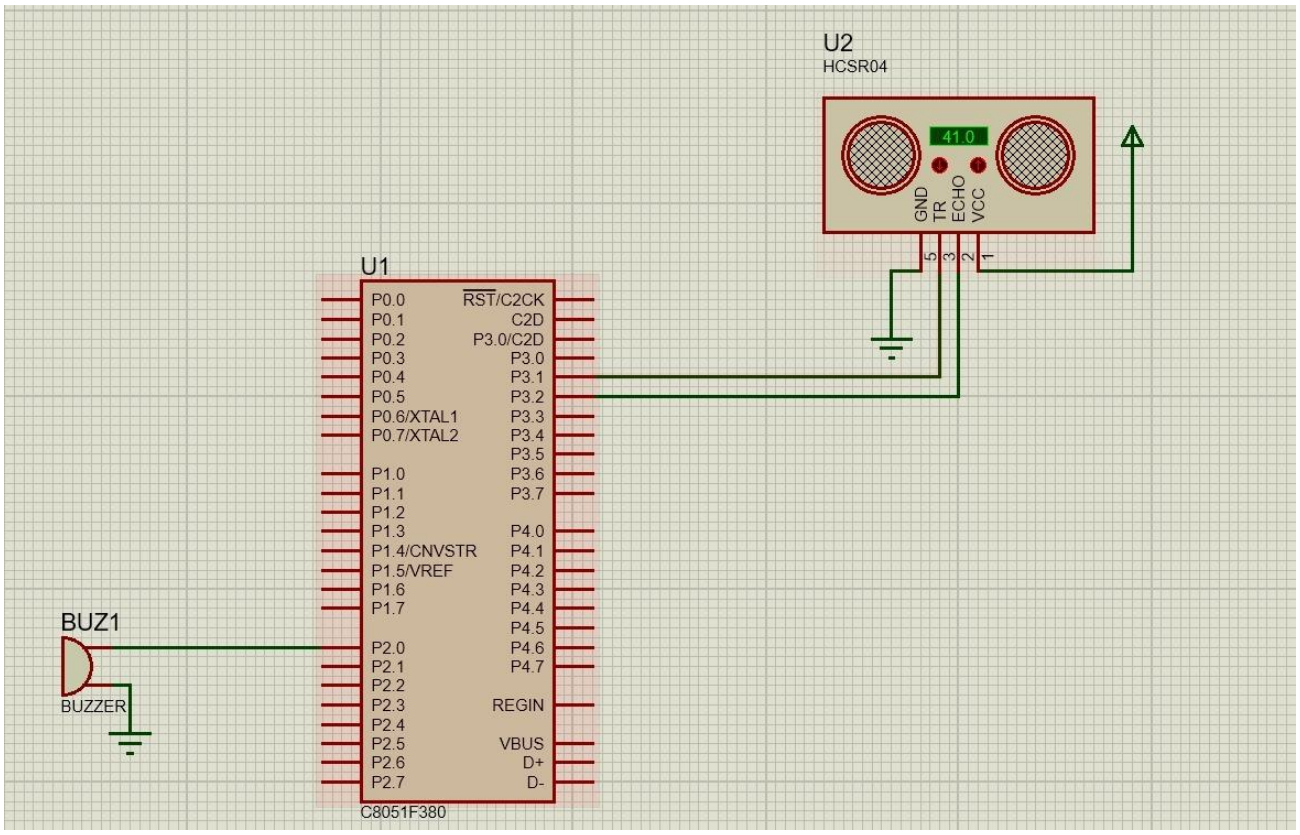
sound, electric bell, and other different sounds.
Typical applications include siren, alarm device, fire alarm, air defense alarm, burglar alarm timer etc.

(2.4) **Interfacing Diagram**



3. Software Design

Proteus Circuit Diagram



Algorithm:

(1.) Define the sound velocity constant
(2.) Define pins for components
(3.) Create Delay_us to create 10us of delay
(4.) Create timer initialization function for timer
(5.) Create trigger pulse sending function
(6.) Create arbitrary delay function
(7.) Create buzzer sounding function
(8.) Send trigger pulse
(9.) Wait for echo pulse, and when it is received, convert distance to centimeters
(10.) If distance is less than 40 cms, then activate buzzer function
(11.) If not, stop the buzzer

Embedded C code

```
#include "c8051f340.h"
#include <math.h>

#define sound_velocity 34300                    /* sound velocity in cm per second */
#define clock_period pow(10,-6)                 /* period for clock cycle of
                                                   8051f340*/

sbit Trigger_pin=P3^1;                  /* Trigger pin */
sbit Echo_pin=P3^2;                     /* Echo pin */
sbit buz = P2^0;                        /* Buzzer pin */
void delay(int);

void Delay_us()
{
        TL0=0xF5;
        TH0=0xFF;
        TR0=1;
        while (TF0==0);
        TR0=0;
        TF0=0;
}

void init_timer()
{
        TMOD=0x01;                      /* Initialise Timer */
        TF0=0;
        TR0=0;
}

void send_trigger_pulse()
{
        Trigger_pin = 0;
        Delay_us();
        Trigger_pin= 1;                 /* pull trigger pin HIGH */
        Delay_us();                     /* provide 10uS Delay*/
        delay(100);
        Trigger_pin = 0;                /* pull trigger pin LOW*/
}
```

```
void delay(int n)
{
double i,j;
for(i=0;i<n;i++);
for(j=0;j<1275;j++);
}


void buzz()                                     /* making buzzer work */
{
        buz = 1;
        delay(100);
        buz = 0;
        delay(100);
}
void main()
{
        int distance_measurement, value;
        XBR1    = 0x40;                         /* Enable Crossbar */
        init_timer();                           /* Initialise Timer*/
   while(1)
        {
                send_trigger_pulse();                   /* send trigger pulse of 10us */

                while(!Echo_pin);               /* Waiting for Echo */
                TR0 = 1;                        /* Timer Starts */
                while(Echo_pin);                        /* Waiting for Echo goes LOW */
                TR0 = 0;                        /* Stop the timer */

                /* calculating distance using timer */
                value = clock_period * sound_velocity;
                distance_measurement = (TL0|(TH0<<8));
                                                        /* read timer register for time count */
                distance_measurement = (distance_measurement*value)/2.0;
                                                        /* find distance(in cm) */

                if(distance_measurement<=40)
                {
                        buzz();
```

```
                }
                else
                        buz = 0;

                delay(100);
        }
}
```

4. Conclusion

By adopting the reverse parking sensor, we have successfully gotten the output of the buzzer if the object is less than 40 cm from the vehicle. While it is not completely possible to be devoid of getting into accidents while parking in reverse, they are going to be prevented to some extent. One thing to take care of is to not have device get ruined due to mechanical or water damage.

5. References

- https://embetronicx.com/tutorials/microcontrollers/8051/ultrasonic-sensor-interfacing-with-8051/
- https://www.quisure.com/blog/faq/what-is-a-buzzer
- https://www.piborg.org/sensors-1136/hc-sr04
- Chandni Patel, Monalisa Swami, Priya Saxena and Sejal Shah," Car parking system", International Journal of Engineering Science and Innovative Technology (IJESIT), Volume 4, Issue 2, March 2015.'
- V. Patil, S. Sawant, P. Phodkar, A. Arkade, S. Kamble and A. Nalawade, "Reverse Car Parking Sensor", International Research Journal of Engineering and Technology (IRJET), Volume 8, Issue 4, April 2021.
- Mazidi, M. A. (2004). The 8051 Microcontroller and Embedded Systems.

Data Sheet (Referred Pages)

C8051F340/1/2/3/4/5/6/7/8/9/A/B/C/D

SFR Definition 21.2. TMOD: Timer Mode

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x89

Bit7: GATE1: Timer 1 Gate Control.
0: Timer 1 enabled when TR1 = 1 irrespective of $\overline{INT1}$ logic level.
1: Timer 1 enabled only when TR1 = 1 AND $\overline{INT1}$ is active as defined by bit IN1PL in register INT01CF (see SFR Definition 9.13).

Bit6: C/T1: Counter/Timer 1 Select.
0: Timer Function: Timer 1 incremented by clock defined by T1M bit (CKCON.3).
1: Counter Function: Timer 1 incremented by high-to-low transitions on external input pin (T1).

Bits5–4: T1M1–T1M0: Timer 1 Mode Select.
These bits select the Timer 1 operation mode.

T1M1	T1M0	Mode
0	0	Mode 0: 13-bit counter/timer
0	1	Mode 1: 16-bit counter/timer
1	0	Mode 2: 8-bit counter/timer with auto-reload
1	1	Mode 3: Timer 1 inactive

Bit3: GATE0: Timer 0 Gate Control.
0: Timer 0 enabled when TR0 = 1 irrespective of $\overline{INT0}$ logic level.
1: Timer 0 enabled only when TR0 = 1 AND $\overline{INT0}$ is active as defined by bit IN0PL in register INT01CF (see SFR Definition 9.13).

Bit2: C/T0: Counter/Timer Select.
0: Timer Function: Timer 0 incremented by clock defined by T0M bit (CKCON.2).
1: Counter Function: Timer 0 incremented by high-to-low transitions on external input pin (T0).

Bits1–0: T0M1–T0M0: Timer 0 Mode Select.
These bits select the Timer 0 operation mode.

T0M1	T0M0	Mode
0	0	Mode 0: 13-bit counter/timer
0	1	Mode 1: 16-bit counter/timer
1	0	Mode 2: 8-bit counter/timer with auto-reload
1	1	Mode 3: Two 8-bit counter/timers