

Project Title - Electronic Pest Repellent

Group no. B01

Shagun Jhaver (06007002) <shaguniitb@iitb.ac.in>

Rahul Singh (06007029) <rahulsingh1@iitb.ac.in>

Tejas Hiremani (06007004) <tejasvph@iitb.ac.in>

Guide: Prof. Girish Kumar

▪ Abstract:

The purpose of the project is to design an ultrasonic pest repellent. Such a device can be very useful to counter the various problems caused by ants, insects, pests, rodents, etc. The device is compact, cheap, and it does not cause any pollution unlike the other chemical repellents. We have used a microcontroller to generate sweep in sound frequencies, and an assembly consisting of audio power amplifier, speaker and LCD for this purpose. The technical details of this project follow later. The circuit has been experimentally tested on ants, bugs, and small insects, and it has been successful in repelling them through the generation of ultrasonic frequency sound.

1. Introduction

It is possible that pests like insects, ants, rats, mice etc. are repelled by ultrasonic frequency in the range of 30 kHz to 50 kHz. Human beings can't hear these high-frequency sounds. Our product repels pests by emitting pulse ultrasonic waves. Using ultrasonic waves creates a noisy and hostile environment which repels pests, whilst remaining absolutely safe for humans and household animals. Unfortunately, all pests do not react at the same ultrasonic frequency. While some pests get repelled at 35 kHz, some others get repelled at 38 to 40 kHz or even higher frequencies. Thus to increase the effectiveness, frequency of ultrasonic oscillator has to be continuously varied between certain limits. Frequency of emission of ultrasonic sound is continuously varied by our product in different patterns to repel different insects.

1.1 Biological Background

Electronic pest control is the name given to the use of any of the several types of electrically powered devices designed to repel or eliminate pests, usually rodents or insects. There are basically two types of electronic pest control devices widely available, these are Ultrasonic and Electromagnetic.

Ultrasonic devices operate by emitting short wavelength, high frequency sound waves too high in pitch to be heard by the human ear (all frequencies greater than 20,000 Hz). This is due to limitations in human hearing. Humans cannot hear ultrasound because the eardrum does not vibrate fast enough, but some animals such as dogs, bats and rodents can hear well into the ultrasonic range. Some insects, such as grasshoppers and locusts can detect frequencies from 50,000 Hz to 100,000 Hz, and moths and lacewings can detect ultrasound as high as 240,000 Hz produced by insect-hunting bats.

Insects detect sound by special hairs or sensilla located on the antennae (mosquitoes) or genitalia (cockroaches), or by more complicated tympanal organs (grasshoppers, locusts, moths and butterflies).

"Ultrasound and Arthropod Pest Control"[1] an extensive Kansas State University study confirmed that ultrasonic sound devices do have both a repellent effect as well as a reduction in mating and reproduction of various insects. However, the results were mixed and ultrasonic sound had little or no effect on some pests. Various ultrasonic devices were highly effective on crickets while the same devices had little or no repellent effect on cockroaches. Additionally the results were mixed with some devices being effective while others having no effect depending on the test subject. The study also concluded there was no effect on ants or spiders in any of the tests. They concluded, based on the mixed results, that more research is needed to improve these devices.

A 2002 study by Genesis Laboratories Inc. does lend some credence to the ability of electronic repellent devices to repel certain pests in controlled environments. "Preliminary study of white-footed mice behavior in the test apparatus demonstrated a significant preference for the non-activated chamber among both sexes."

Cockroaches initially respond to electronic pest control devices by moving about a bit more than usual, but don't appear overly eager to escape from the sound waves. This includes devices that emit uniform frequency as well as changing frequencies of ultrasound. Rodents adjust to the ultrasound (or any new sound) and eventually ignore it. However, researchers were able to use the increased cockroach activity to good effect by increasing the rate they caught the roaches in sticky traps.

Tests of commercial ultrasonic devices have indicated that rodents may be repelled from the immediate area of the ultrasound device for a few minutes to a few days. Other tests have shown that the degree of repellence depends on the frequency, intensity, and the pre-existing condition of the rodent infestation. The intensity of such sounds must be so great that damage to humans or domestic animals would also be likely. Commercial ultrasonic pest control devices do not produce sounds of such intensity.

2. Design Approach

We are using an LM 380 audio power amplifier circuit to design the system capable of producing sound in the frequency range of upto 80 kHz. A speaker of appropriate frequency range is used to transmit these sound waves. We are using a separate power module to power the system. We are using the Atmega-16 microcontroller to produce the different patterns of frequencies which we require in our experimentations, and an LCD – keyboard assembly to track and control this ongoing process. We have programmed the Atmega-16 microcontroller so that it generates the different patterns of frequency sweeps in its different modes. A

keyboard is also provided in the system, so that any of these different modes can be selected by the user.

3. Design of Circuit

3.1 Block Diagram:-

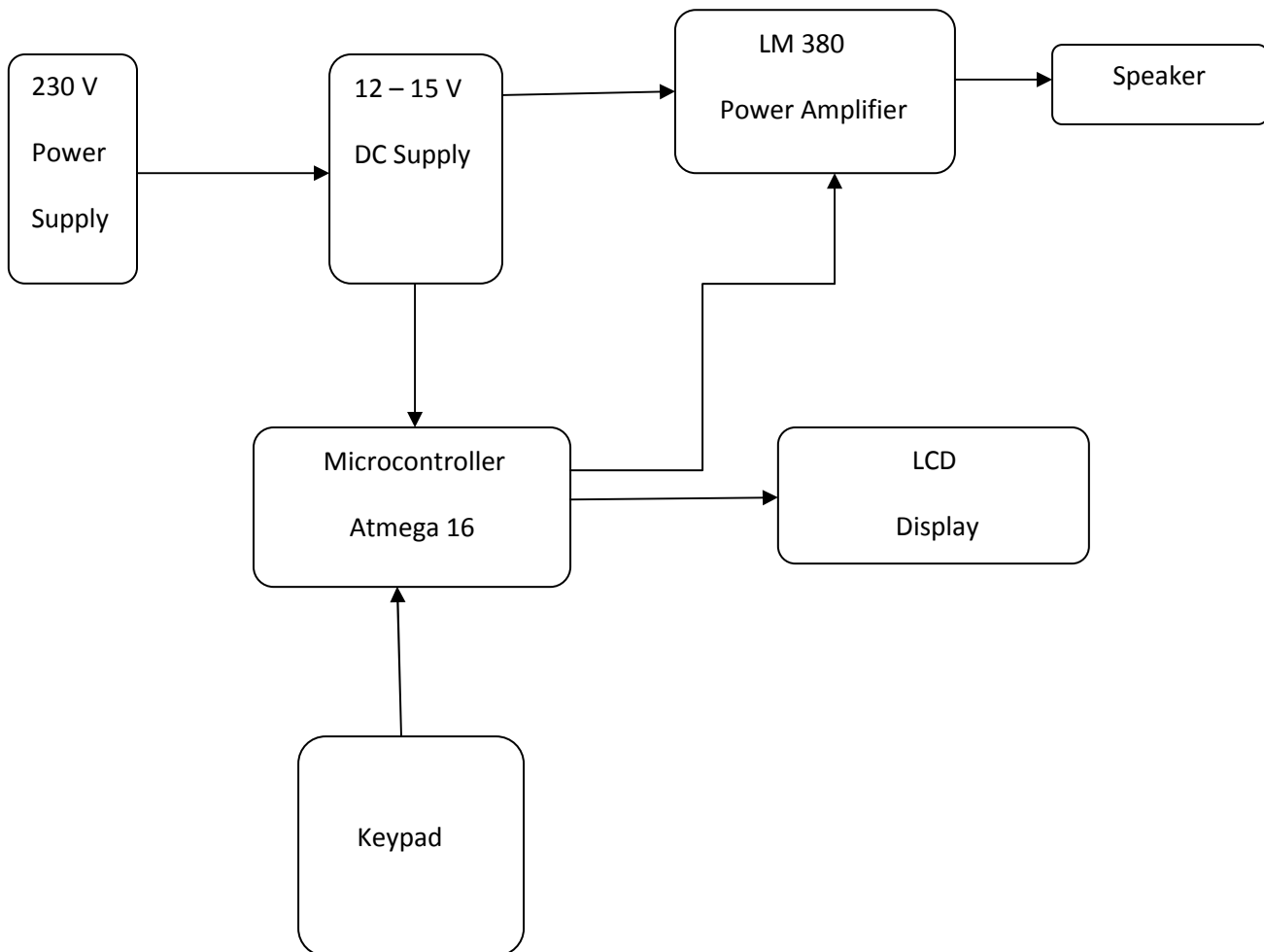


Figure 1 - The above block diagram describes the basic layout of our circuit.

3.2 Main Components:-

3.2.1) LM 380 Ultrasonic Transducer

We are using LM 380 ultrasonic transducer as the amplifier of our circuit. This is a 2.5 W audio power amplifier. Its gain is internally fixed at 34 dB. Some of its important features are :-

- Supply voltage - 10V to 22 V
- Peak current - 1.3 A
- Minimum output power (rms) - 2.5 W ($R_L = 8\Omega$; THD = 3 %)
- A_v (Gain) - 40 – 60 V/V
- Typical A_v - 50 V/V
- Z_{in} (Input Resistance) - 150 k Ω
- Bandwidth - 100 kHz

▪ LM 380 Connection diagram :-

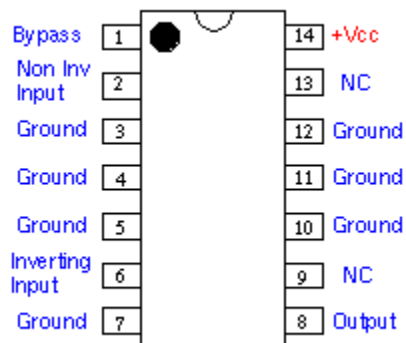


Fig. 2 - LM380 Pin Diagram

▪ LM 380 Block Diagram:-

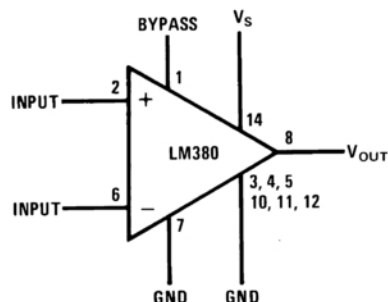


Fig. 3 - LM 380 Block Diagram

▪ **LM 380 Schematic Diagram:-**

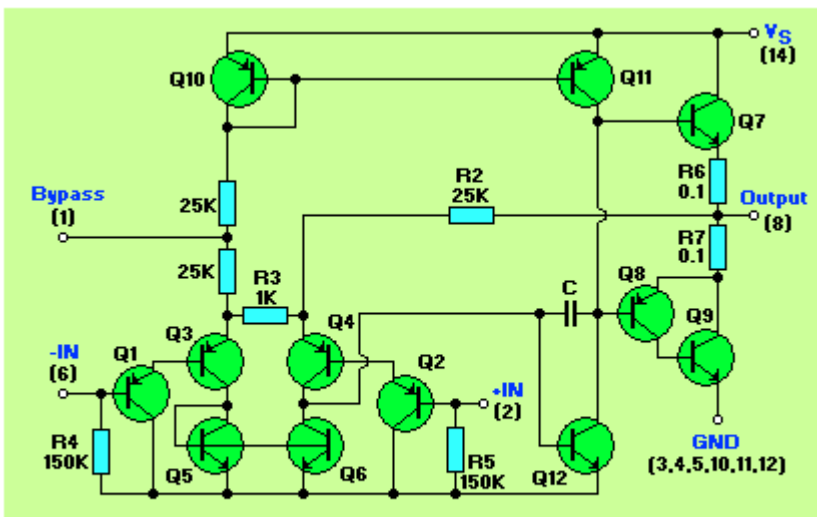


Fig. 17-- Schematic for the LM380 power amplifier from National Semiconductor. It has a 2-watt output rating.

Fig. 4 - LM 380 Schematic Diagram.

3.2.2) Speaker:-

We are using an 8Ω speaker. We have tested the speaker, and its working properly till the frequencies of 100 kHz.

3.2.3) Atmega 16 Microcontroller :-

Atmega 16 is a high-performance, low-power AVR® 8-bit Microcontroller. Its important features are :-

- Advanced RISC Architecture
- 131 Powerful Instructions – Most Single-clock Cycle Execution
- 32 x 8 General Purpose Working Registers
- Fully Static Operation
- Up to 16 MIPS Throughput at 16 MHz
- On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
- JTAG (IEEE std. 1149.1 Compliant) Interface

- Operating Voltages – 2.7 - 5.5V for ATmega16L and 4.5 - 5.5V for ATmega16
- Speed - 0 - 8 MHz for ATmega16L and 0 - 16 MHz for ATmega16

- Atmega 16 Pin Configuration :-

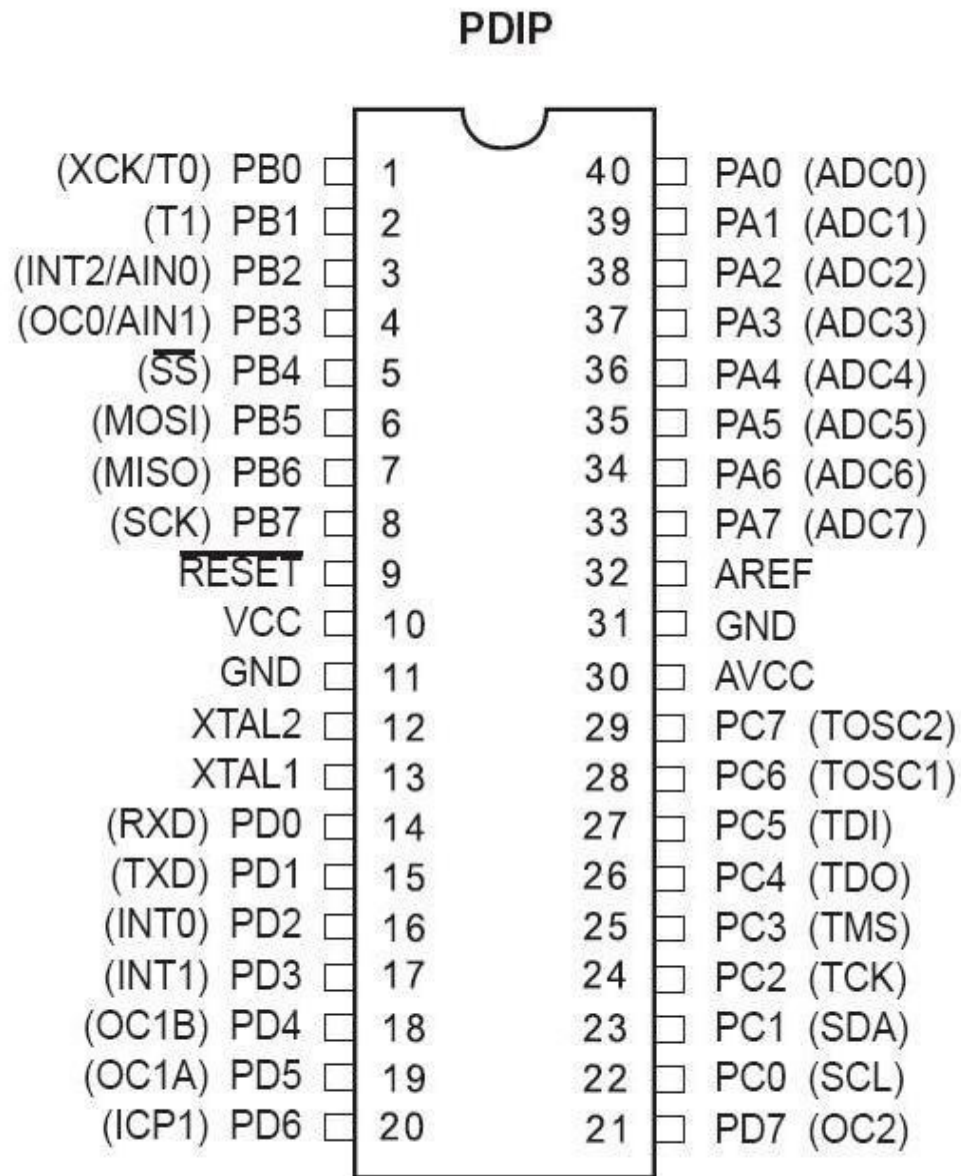


Fig. 5 -Atmega 16 pin configuration

We are using this microcontroller for our circuit, because we need to have different modes of frequency variations, and this serves our purpose. This is also cheap and easily available as compared to some other microcontrollers.

3.2.4) LM 7805 Voltage Regulator: -

The LM78XX series of three terminal regulators is available with several fixed output voltages making them useful in a wide range of applications. One of these is local on card regulation, eliminating the distribution problems associated with single point regulation. The voltages available allow these regulators to be used in logic systems, instrumentation, HiFi, and other solid state electronic equipment. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.

In our circuit, we use this as an input to the microcontroller (as a source).

➤ Voltage Range: -

LM7805C	5V
LM7812C	12V
LM7815C	15V

➤ Features : -

- Output current in excess of 1A
- Internal thermal overload protection
- No external components required
- Output transistor safe area protection
- Internal short circuit current limit
- Available in the aluminum TO-3 package

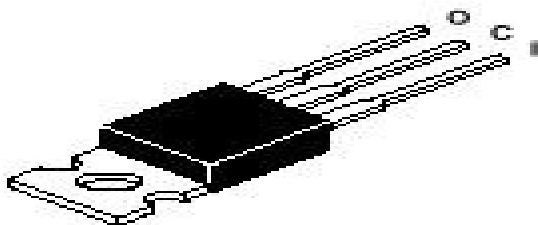
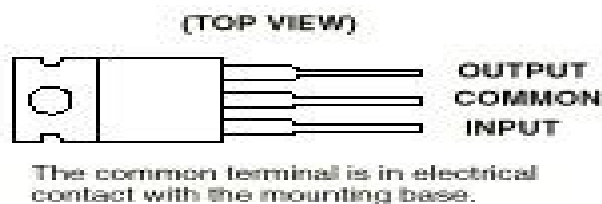


Fig. 6 – LM 7805 top view

3.2.5) Keypad

We are using the 4x4 standard keypad to give input to the microcontroller.

4. Circuit Diagrams and Working Description:-

The circuit can be divided into these basic components;

4.1) Power Source Converter: It converts a 230V, 50 Hz supply into a 12-15 V DC supply. The circuit diagram is shown below;

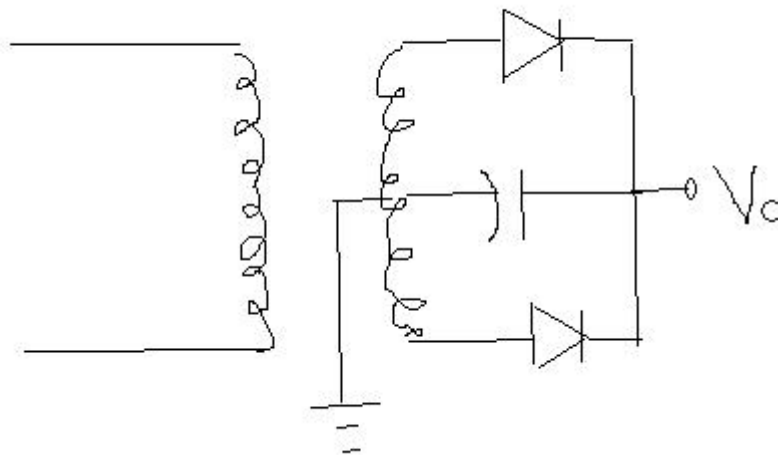


Fig. 7 – Power Source Converter

It uses a centre tap transformer followed by a rectifying diode circuit. The output is taken across the capacitor.

4.2) Audio Power Amplifier: The audio power amplifier takes 1 V p-p square wave input generated from the microcontroller unit and gives an amplified signal to the speaker. The LM380 IC used is a 2.5 W audio amplifier. The output of the amplifier was measured using an ultrasonic receiver circuit during testing stages, and the gain was found to remain almost constant upto 80 KHz, a range conveniently suited to our needs. The circuit diagram of the amplifier circuit is shown below :

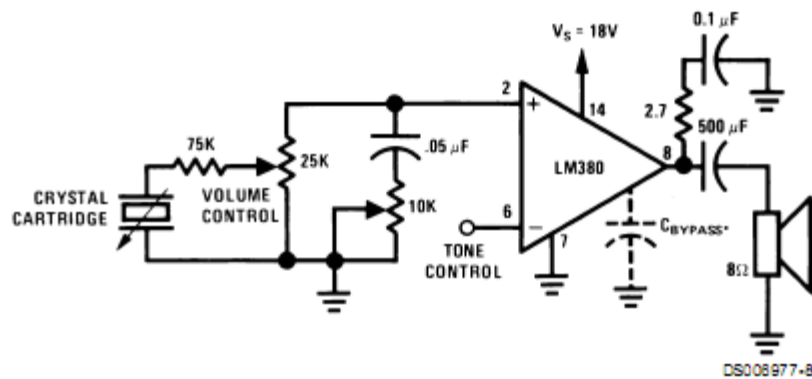


Fig. 8 – Phono Amplifier Circuit Diagram

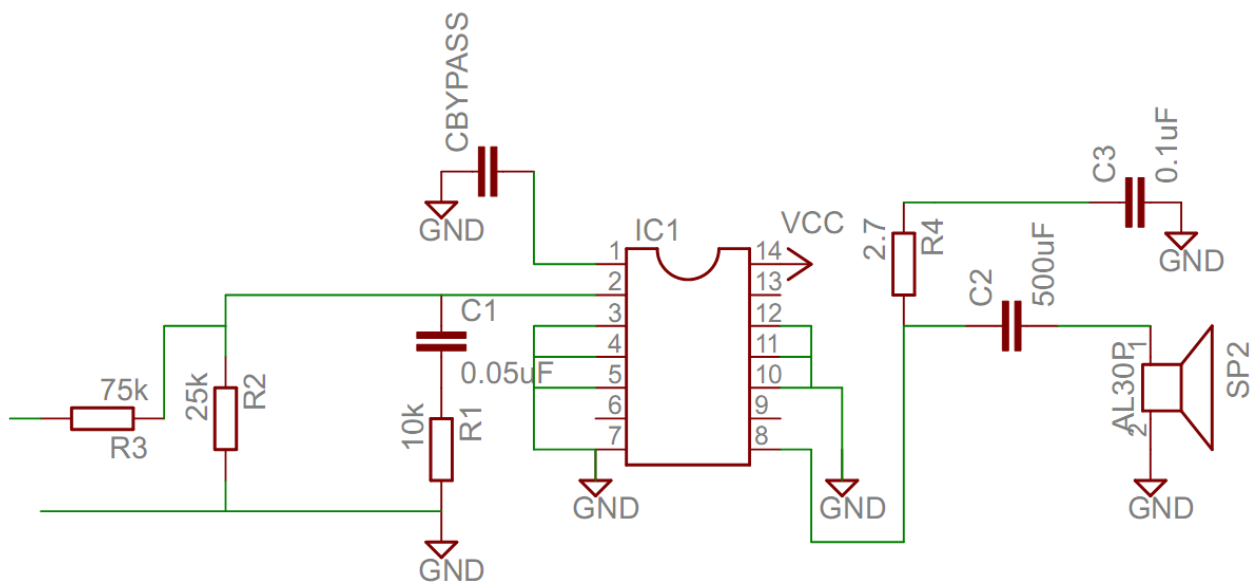


Fig.9 – Phono Amplifier Circuit Schematic Diagram.

4.3) Microcontroller Unit: The microcontroller unit can operate in various modes depending on user input. In each mode the microcontroller generates a square wave signal at portD whose frequency varies continuously in a set range. This ensures that the sound output continuously changes and the pest doesn't get used to the sound. The various segments of the microcontroller unit are:

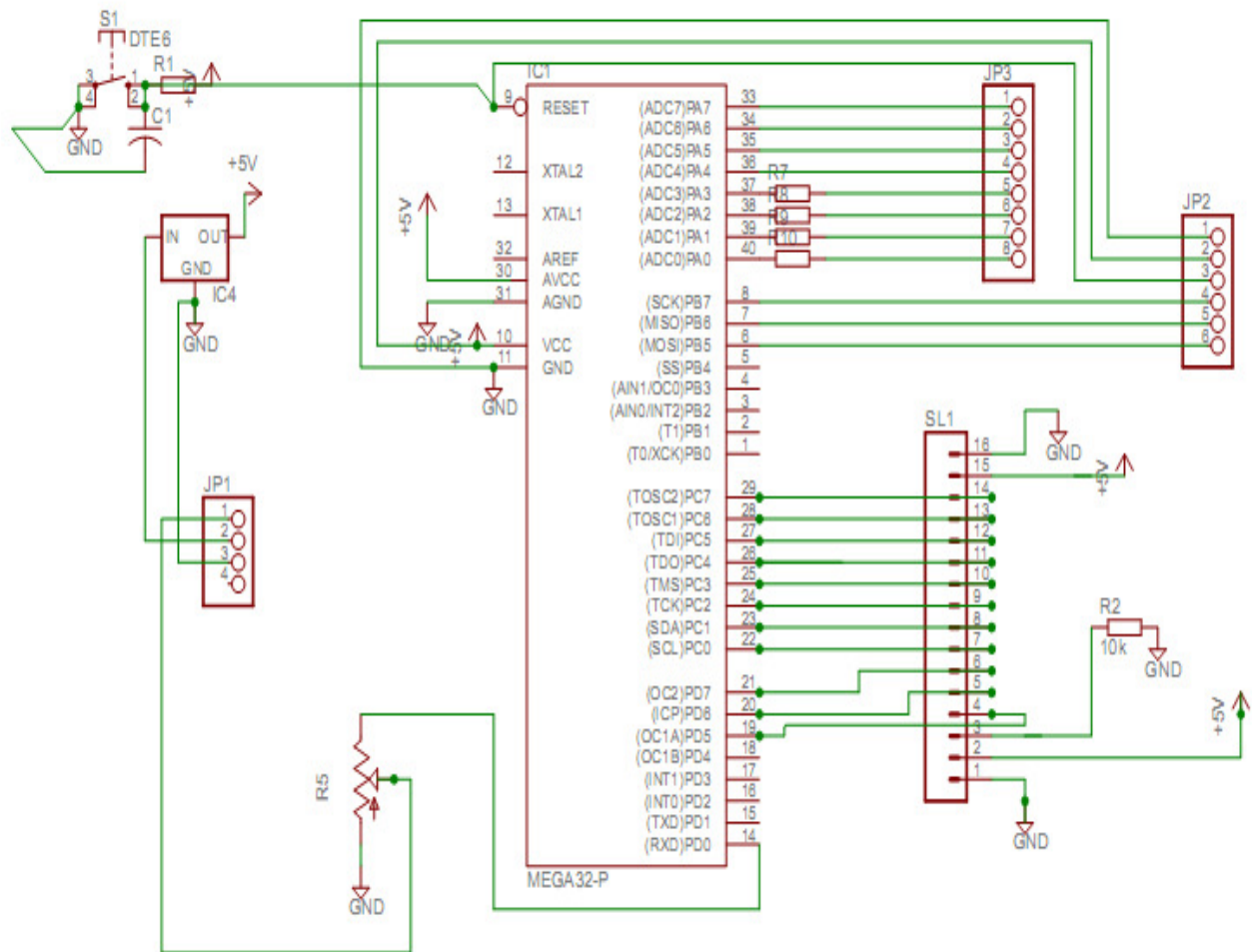


Fig. 10-microcontroller circuit diagram

4.4) 4x4 Keypad: The method of polling the 4x4 keyboard is as follows:

In general, keyboards are organized as a matrix of rows and columns; two side of this matrix are connected to V_{CC} through resistors while the third side is connected to the microcontroller port and configured as an output; and the last side is connected to the microcontroller port and configured as an input as shown in fig. 1.

Microcontroller keep scanning the keyboard, when all inputs are high ("ones") that means no key is pressed; if one bit is low ("zero"), that means there is a pressed key. System

designer setup a Look-Up Table contain the ASCII code for each key, in this project we will use 16 keys to represent the hex number from 0 to F arranged according to keyboard arrangement.

To detect which key is pressed; microcontroller ground all rows, then reads all columns, if all the columns=1's no key is pressed, if one columns=0 it's indicate that a key is pressed. To identify the exact pressed key, microcontroller will start with the top row by grounding it; and then read the columns. If the data read is all ones, no key is row activated, and the process will move to the next row, until it reaches the row that has a pressed key. At this stage, microcontroller knows the row that has a pressed key, and can setup the starting address in the look-up table for that row.

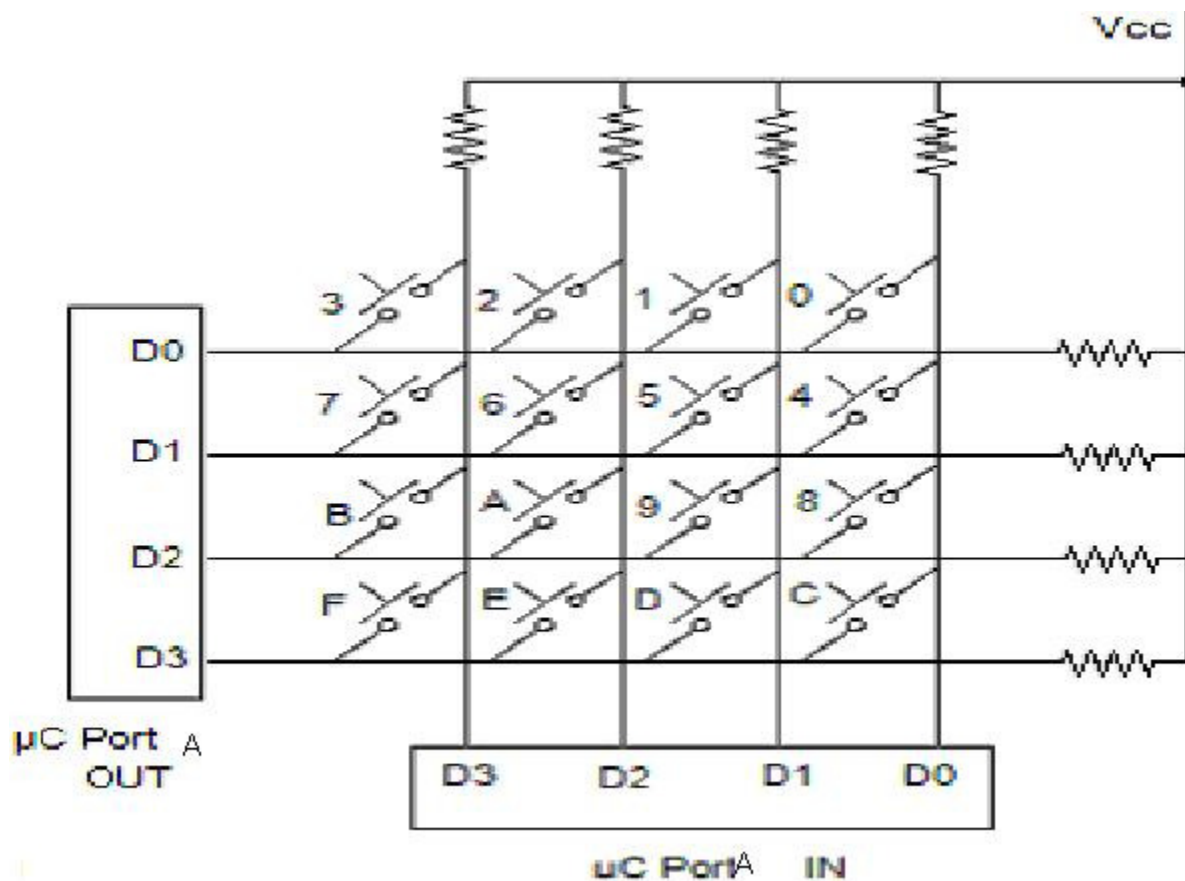


Fig. 11 – 4x4 Keypad pin diagram

The last step is finding the column that has a pressed key by rotating the columns bits; one bit at a time to locate a low bit, the most efficient way is rotating column bits through the carry flag by using RRC instruction. When the 0 bit is found, microcontroller pulls the corresponding code from the look-up table.

To be sure that no key is still down from the previous session, microcontroller send 0's to all rows at on time and check the columns, if all columns are high then the normal scanning start, otherwise, it will wait until all columns become high.

4.5) LCD Display: We use a standard 16 pin LCD display to view the choice entered by the user and the current mode in which the circuit is working. The pin configuration of the LCD used is shown below:

- 1.Vss - Gnd
- 2.Vcc - +5V
- 3.Vee - Contrast control
4. RS - Register select [Command / Data]
5. RW - Read / Write
6. EN - Enable pulse.
7. D0 - Data bus LSB
8. D1
-
-
14. D7 - Data bus MSB.
- 15 & 16 - Backlight.

4.6) Frequency generation: We use timer in CTC mode to generate the square wave. The timer interrupts are enabled. So on every timer compare with the OCR1A value, the internal interrupt is triggered which toggles the D0 bit. To enable continuous variation of frequency the value of the OCR1A register is continuously changed after 10000 pulses. This is included in the ISR routine of the timer. The final code used in the microcontroller is attached in the appendix.

5) Testing and demo:

We first carried out the tests on ants and some bugs. They appeared to stray away around the frequencies of 5 kHz upto 35 kHz, and they got agitated when the circuit was turned on. But it was very difficult to get a clear view, because biological systems are not as predictable as normal electrical systems, so we took an educated guess after a number of trials.

5.1) Test on Rats:

We tested the circuit on a small rat, which was within an enclosed metallic cage. The problem is that when the rat moves around the cage, the natural vibrations of the cage itself are in the ultrasonic range. When we turned on our circuit, the rat in some cases, would move away from the source of the noise, in other cases, it would 'freeze', where it would not move at all. This was around the frequencies of 40 – 45 kHz. Apparently, the rat would get irritated, when the circuit was turned on. We also found on internet that rats are repelled at around 40-45 kHz.

6) Problems faced:

➤ *Heat sink*

The LM 380 IC was getting heated up in the phono amplifier circuit. So, for the PCB design, we made copper squares in order to remove the heat.

➤ *Polling*

Polling is a continuous process and the frequency generation is also a continuous process. So, before we knew about timer interrupts, doing both of them simultaneously was a problem

7) Suggestions for further improvement:

➤ *Bigger speakers.*

Speakers better suited in the ultrasonic range could have been used.

➤ *Standard product*

We had ordered an ultrasonic pest repellent product from Varna Enterprises, Bangalore to get a look at the commercial products currently available in the market using this technology. Quite surprisingly, when we opened up and analyzed the circuit for this product, we found it fairly simple. We could not test this product on any pests, etc. because it was broken during the couriering process, and wasn't working. We mailed the product back to get a new item of the same product, but the Varna Enterprises made a lot of delay, and we would not be able to get the new item on the date of EDL evaluation.

➤ *Combined method of using ultrasonic waves, chemicals, electromagnetic and ionic technique.*

We can use all the three techniques together to make the product much more reliable. However, using chemicals/ionic repellents makes the product no more pollution free and is not an electrical solution. Electromagnetic pest repelling devices claim to alter the electromagnetic (EM) field of household wiring, and vibrate the electromagnetic field that always exists around the wiring in the walls of your house but they have been banned in Canada because of health reasons. So, we have only used ultrasonic technique In our product.

References

http://en.wikipedia.org/wiki/Electronic_pest_control

<http://www.electronicpestrepeller.com/>

http://sharada.ee.iitb.ac.in/~wel12/Components%20Records/analog%20ic_datasheet/lm380.pdf

http://sharada.ee.iitb.ac.in/~wel12/Components%20Records/analog%20ic_datasheet/78xx.pdf

http://www.alibaba.com/product-gs/50518372/Pest_Repeller.html

<http://www.goodlifecompany.com/shopexd.asp?id=25>

http://www.pestrepellerultimate.com/pest_repeller_ultimate_sg.htm

Appendix

A.1) User Manual:

We just need to plug the unit into the ac supply and it starts working. The LCD Display shows a hello sign. There are five modes in the unit;

Mode 0: 20-30KHz (Suitable for little bugs,mosquitoes etc)

Mode 1; 30-40KHz (Suitable for rats)

Mode 2: 40-60KHz (Suitable for bats, cockroaches etc)

Mode 3: General Mode; Range 20-60KHz

Mode 4: Audio Mode; 100Hz-10KHz

Mode 'X' Sign on the LCD shows that the choice was not registered while Mode '!' shows that no such mode exists.

There is also a reset button to reset the system.

A.2) We used the following code for the microcontroller:-

```
#include<avr/io.h>

#include<util/delay.h>

#include<avr/interrupt.h>

// *****LCD Functions*****

#define dataout (PORTC)

#define datain (PINC)

#define enable (PD7)

#define rw (PD6)

#define rs (PD5)
```

```
#define KB_PORT_OUT (PORTA)
```

```
#define KB_PORT_IN (PINA)
```

```
int q=0;
```

```
void wait(void)
```

```
{
```

```
PORTD &= ~_BV(enable);
```

```
PORTD &= ~_BV(rs); // changing the values for reading the resistor values
```

```
PORTD |= _BV(rw);
```

```
dataout = 0x00;
```

```
DDRC= 0x00;
```

```
PORTD |= _BV(enable);
```

```
    if ((PORTC & _BV(PC7)) != 0x00)
```

```
    {
```

```
        wait();
```

```
    }
```

```
PORTD &= ~_BV(enable);
```

```
PORTD &= ~_BV(rw);
```

```
dataout = 0xff;
```

```
DDRC=0xff;
```

```
}
```

```
void init_lcd (void)
```

```
{  
  
PORTD &= ~_BV(enable);  
  
PORTD &= ~_BV(rw) & ~_BV(rs);  
  
dataout = 0x38;  
  
PORTD |= _BV(enable);  
  
_delay_ms(2);  
  
PORTD &= ~_BV(enable);  
  
wait();  
  
  
PORTD &= ~_BV(rw) & ~_BV(rs);  
  
dataout = 0x0e;  
  
PORTD |= _BV(enable);  
  
_delay_ms(2);  
  
PORTD &= ~_BV(enable);  
  
wait();  
  
  
PORTD &= ~_BV(rw) & ~_BV(rs);  
  
dataout = 0x06;  
  
PORTD |= _BV(enable);  
  
_delay_ms(2);  
  
PORTD &= ~_BV(enable);  
  
wait();  
  
}
```

```

void clear_lcd(void)

{

PORTD &= ~_BV(rw) & ~_BV(rs);

dataout = 0x01;

PORTD |= _BV(enable);

_delay_ms(2);

PORTD &= ~_BV(enable);

_delay_ms(2);

}

```

```

void print_lcd(unsigned char x)           // function to print the given value in ASCII

{

PORTD &= ~_BV(rw);

PORTD |= _BV(rs);

dataout = x;

PORTD |= _BV(enable);

_delay_ms(2);

PORTD &= ~_BV(enable);

wait();

}

```

```

//*****END OF LCD Part*****

```

```

void port_init(void)

```

```

{
    DDRB = 0x00;

    PORTB = 0x00;

    DDRA = 0x0f; //Key-board port, higer nibble - input,lower nibble output

    PORTA = 0xff; //pull-up enabled for higher nibble

    DDRC = 0xff;

    //PORTC = 0x00;

    DDRD = 0xff;

    //PORTD = 0x00;

}

```

```

int check(void)
{
    unsigned char upperNibble1;

    upperNibble1 = 0xff;

    int f=0;

    for(int i=0; i<4; i++)
    {
        _delay_ms(1);

        KB_PORT_OUT = ~(0x01 << i);

        _delay_ms(1); //delay for port o/p settling

        upperNibble1 = KB_PORT_IN | 0x0f;

        if (upperNibble1 != 0xff)

```

```

{
    _delay_ms(20); //key debouncing delay

    upperNibble1 = KB_PORT_IN | 0x0f;

    if(upperNibble1 != 0xff)

        f=1;

}

}

return(f);

}

```

```

ISR(TIMER1_COMPA_vect)

```

```

{

    PORTD = PORTD ^ 0x01; // Toggle the d0 bit

    q=q+1;

    if(q=10000)

    {

        OCR1A = OCR1A-1;

        q=0;

    }

}

```

```

//***** MAIN FUNCTION *****

```

```

int main(void)

```

```
{  
  
port_init();  
  
float a;  
  
double z;  
  
unsigned char upperNibble, keyCode, keyPressed, i;  
  
  
//initialising the lcd  
  
init_lcd ();  
  
clear_lcd();  
  
print_lcd(0x48);  
  
print_lcd(0x45);  
  
print_lcd(0x4C);  
  
print_lcd(0x4C);  
  
print_lcd(0x4F);  
  
  
//initialising the timer  
  
TCCR1B |= (1 << WGM12); // Configure timer 1 for CTC mode  
  
  
TIMSK |= (1 << OCIE1A); // Enable CTC interrupt  
  
  
sei(); // Enable global interrupts  
  
  
OCR1A = 50; // Set CTC compare value to 1Hz at 1MHz AVR clock, with a prescaler of 1
```

```
TCCR1B |= (1 << CS10);
```

```
// Key polling part
```

```
while(1)
```

```
{
```

```
upperNibble = 0xff;
```

```
for(i=0; i<4; i++)
```

```
{
```

```
_delay_ms(1);
```

```
KB_PORT_OUT = ~(0x01 << i);
```

```
_delay_ms(1); //delay for port o/p settling
```

```
upperNibble = KB_PORT_IN | 0x0f;
```

```
if (upperNibble != 0xff)
```

```
{
```

```
_delay_ms(20); //key debouncing delay
```

```
upperNibble = KB_PORT_IN | 0x0f;
```

```
if(upperNibble == 0xff) goto OUT;
```

```
RESUME;;
```

```
keyCode = (upperNibble & 0xf0) | (0x0f & ~(0x01 << i));
```

```
while (upperNibble != 0xff)

upperNibble = KB_PORT_IN | 0x0f;


_delay_ms(20); //key debouncing delay


switch (keyCode) //generating key character to display on LCD
{
case (0xee): keyPressed = 0x30;

break;

case (0xed): keyPressed = 0x31;

break;

case (0xeb): keyPressed = 0x32;

break;

case (0xe7): keyPressed = 0x33;

break;

case (0xde): keyPressed = 0x34;

break;

case (0xdd): keyPressed = 0x21;

break;

case (0xdb): keyPressed = 0x21;

break;

case (0xd7): keyPressed = 0x21;

break;

case (0xbe): keyPressed = 0x21;
```

```
break;

case (0xbd): keyPressed = 0x21;

break;

case (0xbb): keyPressed = 0x21;

break;

case (0xb7): keyPressed = 0x21;

break;

case (0x7e): keyPressed = 0x21;

break;

case (0x7d): keyPressed = 0x21;

break;

case (0x7b): keyPressed = 0x21;

break;

case (0x77): keyPressed = 0x21;

break;

default : keyPressed = 0x58;

} //end of switch
```

```
clear_lcd();

print_lcd(0x4D);

print_lcd(0x4F);

print_lcd(0x44);

print_lcd(0x45);

print_lcd(0x20);
```

```
print_lcd(keyPressed);
```

```
_delay_ms(1000);
```

```
OUT;;
```

```
if (keyPressed == 0x30)
```

```
{
```

```
OCR1A = 50;
```

```
while(1)
```

```
{
```

```
if (OCR1A==33)
```

```
OCR1A=50;
```

```
if (check()==1)
```

```
goto RESUME;
```

```
}// end of while
```

```
}//end of if
```

```
if (keyPressed == 0x31)
```

```
{
```

```
OCR1A = 33;
```

```
while(1)
```

```
{
```

```
if (OCR1A==25)
```

```
OCR1A=33;
```

```
if (check()==1)
goto RESUME;
} // end of while
} //end of if
```

```
if (keyPressed == 0x32)
{
OCR1A = 25;
while(1)
{
if (OCR1A==20)
OCR1A=25;
if (check()==1)
goto RESUME;
} // end of while
} //end of if
```

```
if (keyPressed == 0x33)
{
OCR1A = 50;
while(1)
{
if (OCR1A==20)
OCR1A=50;
```

```
if (check()==1)

goto RESUME;

} // end of while

} //end of if
```

```
if (keyPressed == 0x34)

{

OCR1A = 1000;

while(1)

{

if (OCR1A==50)

OCR1A=1000;

if (check()==1)

goto RESUME;

} // end of while

} //end of if
```

```
} //end of if

} //end of for

} //end of while(1)
```

```
return(0);

} //end of main()
```