“ Function Generator ”

[Project Report]

Bachelor of Technology
in
Electronics and Communication Engineering

By : -

Het Patel
[22BEC050]

Hirpara Yagnesh
[22BEC051]

Course : - Microcontroller and Interfacing

Course Code : - 2EC701

Abstract

This project presents the design and implementation of a versatile function generator using a microcontroller 89C52 (8052), along with supporting components such as respack 8, LCD display, keypad, DAC0808, op-amp, capacitors, and resistors. The primary goal of this project is to create a user-friendly device capable of generating various types of waveforms with high accuracy.

The function generator provides four different types of waves: sine, square, triangle, and sawtooth. Users can input the desired frequency and select the waveform type through a keypad interface. The selected waveform and corresponding frequency are displayed on an LCD screen for easy monitoring.

The heart of the system is the microcontroller 89C52, which orchestrates the generation of waveforms by controlling the DAC0808 digital-to-analog converter. The DAC0808 converts digital signals from the microcontroller into precise analog voltages, which are then amplified by an operational amplifier to produce the desired waveforms.

Keywords

Function Generator, Microcontroller 89C52 (8052), Waveform Generation, DAC0808, LCD Display, Keypad Interface, Operational Amplifier, User-friendly, High Accuracy, Electronics Experimentation

Introduction

In the world of electronics, making precise signals is really important. Whether it's for learning or practical use, having a reliable way to create different kinds of waves is crucial. This project is all about creating a Function Generator. It's a device that uses a microcontroller called the 89C52 along with some other parts like respack 8, a screen to show information, a keypad for input, and other components.

The main goal of this project is to make it easy for people to choose what kind of wave they want and how often it repeats (its frequency), and to make sure that the waves are made accurately. By putting together both hardware and software, this Function Generator tries to be something that's affordable and useful for both learning and for people who like to play with electronics.

This project is not just about making a cool gadget, it's also about understanding how signals work in electronics. By doing this project, people can learn a lot about how electronics work, and hopefully, have fun doing it too!

Diagram

This diagram illustrates the overall layout of the function generator circuit, showcasing the interconnection of components such as the microcontroller 89C52, respack 8, LCD display, keypad interface, DAC0808, operational amplifier, capacitors, and resistors.The Circuit is shown in **Figure 1.**

It visually presents how these components are interconnected to facilitate the generation of various waveforms with precise frequency control.

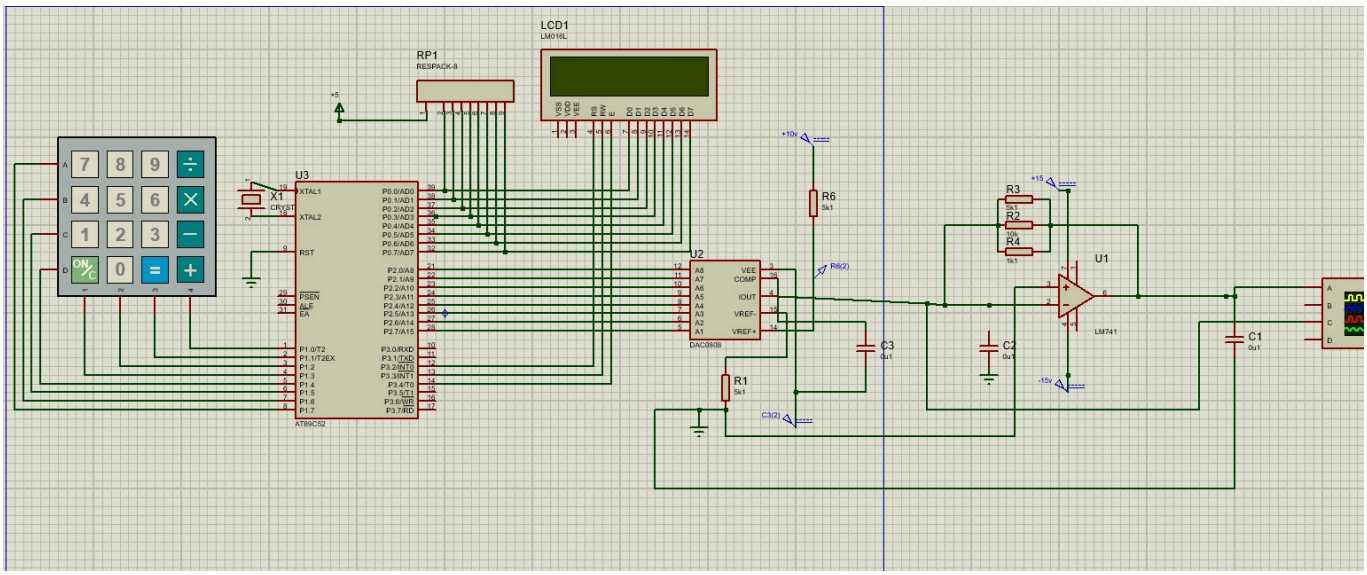

Figure 1

Working

The function generator project operates through a combination of hardware and software components to generate different types of waves with precise frequencies.
Hardware Components:

1. **Microcontroller 89C52:** This serves as the brain of the system, controlling and coordinating the operation of other components. It executes the software instructions to generate digital signals corresponding to the desired waveforms and frequencies.
2. **Respack 8:** This component provides additional memory storage for the microcontroller, enabling it to store and execute the necessary program instructions efficiently.
3. **LCD Display:** The LCD screen serves as the user interface, displaying information such as the selected waveform type and frequency for easy monitoring and interaction.
4. **Keypad Interface:** Users can input the desired frequency and select the waveform type using the keypad interface. This input is processed by the microcontroller to adjust the waveform generation accordingly.
5. **DAC0808:** The Digital-to-Analog Converter (DAC0808) converts the digital signals generated by the microcontroller into precise analog voltages. These analog voltages represent the waveform shapes (sine, square, triangle, sawtooth) based on the digital inputs received.
6. **Operational Amplifier:** The operational amplifier (op-amp) amplifies the analog voltages produced by the DAC0808 to the required levels for waveform generation. It ensures that the output signals accurately reflect the desired waveform shapes and frequencies.
7. **Capacitors and Resistors:** These components are used for signal conditioning and filtering purposes, ensuring the stability and accuracy of the generated waveforms.

Flowchart

Start

User Input (Frequency, Waveform Type)

Display User Instructions on LCD

Get User Input from Keypad

Generate Digital Signal
~ Calculate Digital Signal for Selected Frequency and Waveform Type

Convert Digital Signal to Analog
~ DAC0808 Conversion

Amplify Analog Signal
~ Amplify Signal using Operational Amplifier

Output Signal

End

~ This flowchart illustrates the step-by-step process of the project. This flowchart provides a clear overview of how the project functions, from user input to output signal generation.

Results

Sine Wave:

The sine wave is generated by smoothly varying the voltage level over time, following the mathematical curve of the sine function. This is achieved by incrementally adjusting the output voltage in a sinusoidal pattern using the DAC0808 and amplifying it through the operational amplifier.The same is shown in **Figure 2.**

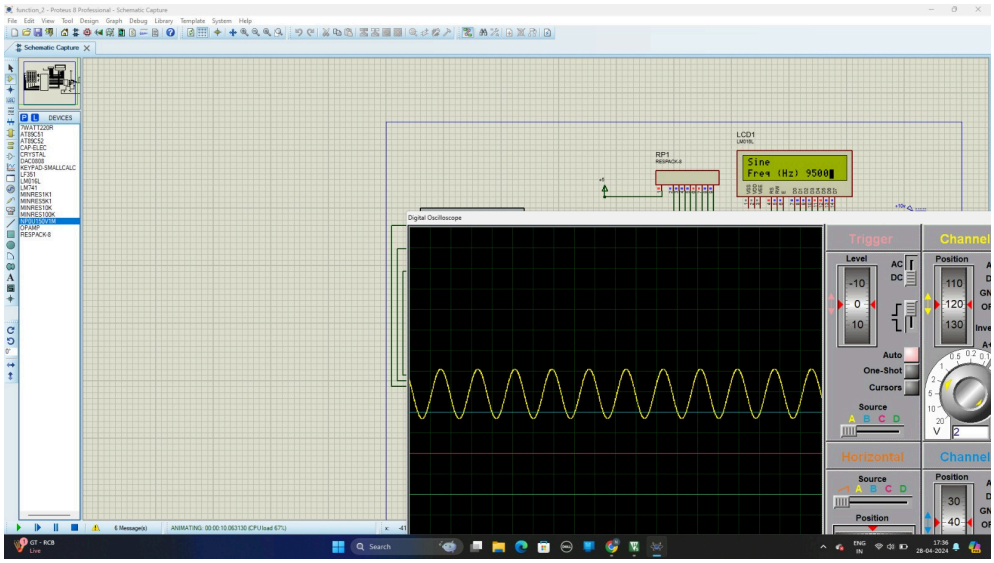

Figure 2

Sawtooth Wave:

The sawtooth wave is generated by rapidly increasing the voltage level to a peak value and then abruptly dropping back to a minimum value, creating a sharp, linear rise followed by a sudden fall. This sawtooth pattern is generated by the microcontroller sending digital signals to the DAC0808, which produces the sawtooth waveform, amplified by the operational amplifier.The same is shown in **Figure 3**.

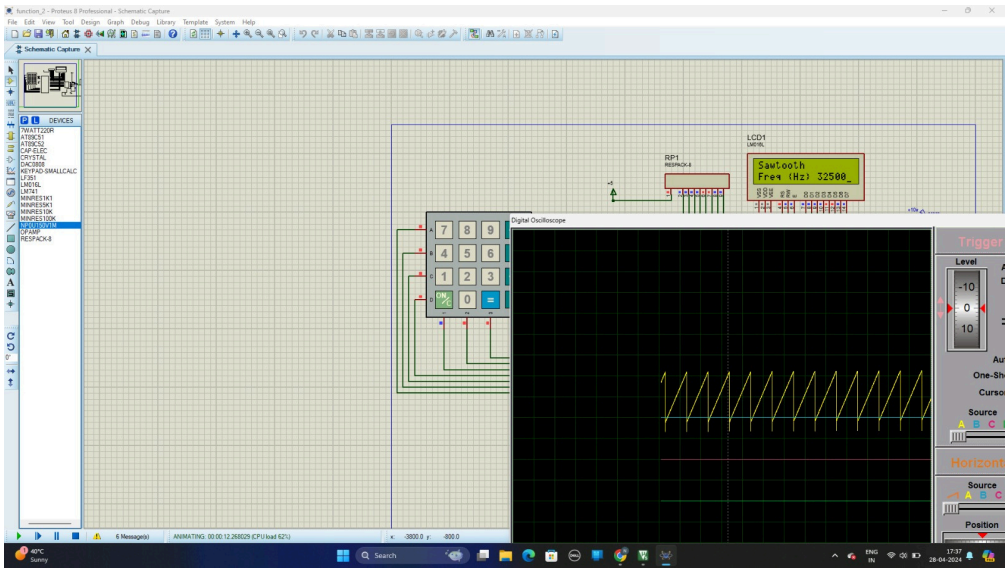

Figure 3

Square Wave:

The square wave is created by alternately switching the output voltage between two levels, typically high and low, in a regular pattern. This on-off switching is controlled by the microcontroller, which sends digital signals to the DAC0808 to produce the square waveform, amplified by the operational amplifier.The same is shown in **Figure 4.**

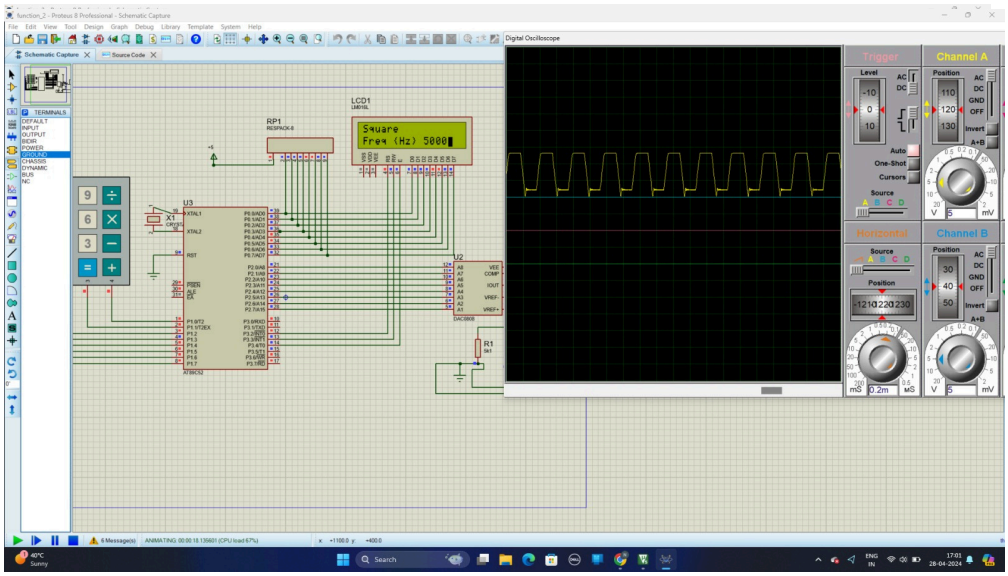

Figure 3

Triangular Wave:

The triangle wave is formed by linearly increasing and decreasing the voltage level over time, resulting in a waveform that rises and falls at a constant rate. This linear voltage ramp is achieved by the microcontroller generating a sequence of digital signals to produce the triangle waveform through the DAC0808, which is then amplified by the operational amplifier.The same is shown in **Figure 4**.

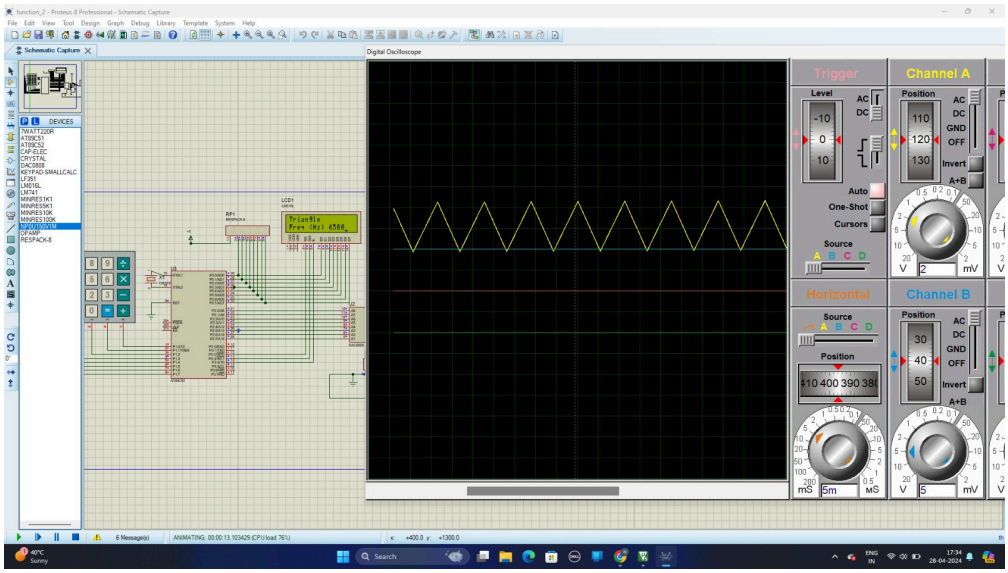

Figure 4

Bill of Materials

Components	Price
uC 89c52	110Rs
DAC0808	90Rs
IC 741	10Rs
Respack 9	40Rs
Crystal Osc.	20Rs
Resistors	50Rs
Capacitors	30Rs
LCD Display	120Rs
4x4 Keypad	40Rs
Jumper wires	25Rs
Total	535Rs

Applications

The function generator project has a wide range of applications in both educational and practical settings. In educational institutions, it serves as a valuable tool for teaching and learning purposes in electronics and electrical engineering courses. Students can use the function generator to study and experiment with different types of waveforms, gaining practical insights into waveform generation, signal processing, and circuit design.

In laboratory settings, the function generator is utilized for various testing and measurement tasks. Engineers and technicians can use it to calibrate and test electronic circuits, analyze the behavior of components under different conditions, and troubleshoot signal processing systems. Its versatility in generating sine, square, triangle, and sawtooth waves makes it suitable for a wide range of applications, including audio signal testing, sensor calibration, and frequency response analysis.

Overall, the function generator project offers a valuable tool with diverse applications in education, research, and industry. Its versatility, accuracy, and user-friendly operation make it an essential asset for electronics enthusiasts, students, engineers, and technicians alike.

Conclusion

In conclusion, the development and implementation of the function generator project have demonstrated the successful integration of hardware and software components to achieve the objective of generating various waveforms with precision and user-friendly operation. By utilizing the microcontroller 89C52 along with supporting components such as the DAC0808, LCD display, and keypad interface, the project has provided a versatile tool for both educational and practical purposes in the field of electronics.

The function generator offers a cost-effective solution with high accuracy in waveform generation. The user-friendly interface allows users to easily select the desired waveform type and frequency, while the accurate generation of waveforms ensures reliable performance in a variety of applications.

Overall, the function generator project not only serves as a valuable educational tool for understanding the principles of waveform generation and signal processing but also showcases the practical application of electronics engineering concepts.

References

[1] Mazidi, Muhammad Ali, Janice Gillispie Mazidi, and Rolin D. McKinlay. "The 8051 microcontroller and embedded systems using assembly and C." Pearson Education India, 2008.

[2] Malik, Mazhar Ali, and Abdul Haseeb Ansari. "8051 Microcontroller: Hardware, Software, and Applications." Oxford University Press, 2019.

[3] "8051 Microcontroller Datasheet." Atmel Corporation.

[4] "DAC0808 Datasheet." National Semiconductor. https://www.ti.com/lit/ds/symlink/dac0808.pdf

[5] Ibrahim, Dogan. "PIC Microcontroller Projects in C: Basic to Advanced." Newnes, 2008.

Source Code

```
#include <reg51.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <intrins.h>
#define display_port P0      // Data pins connected to port 0 on microcontroller


sbit rs = P3^2;              // RS pin connected to pin 2 of port 3
sbit rw = P3^3;               // RW pin connected to pin 3 of port 3
sbit e =  P3^4;              // E pin connected to pin 4 of port 3

sbit C4 = P1^0;     // Connecting to Port 1
sbit C3 = P1^1;
sbit C2 = P1^2;
sbit C1 = P1^3;
sbit R4 = P1^4;
sbit R3 = P1^5;
sbit R2 = P1^6;
sbit R1 = P1^7;
sbit atoi_test = P3^5;
sfr DAC = 0x90; // Port P2 address

char freq_in[6];
long int freq_int = 0;
long unsigned int freq_square = 0;
long unsigned int freq_triangle = 0;
long unsigned int freq_sawtooth = 0;
long unsigned int freq_sine = 0;
char i = 0;

// Constants for waveform types
bit sine_flag =0;
bit square_flag =0;
bit triangle_flag =0;
bit sawtooth_flag =0;
int freq_selected = 0; // Flag indicating whether frequency is selected
int selected_waveform = 0; // Selected waveform type
```

```
void msdelay(unsigned int time) {
   unsigned int i, j;
   for (i = 0; i < time; i++)
      for (j = 0; j < 1275; j++);
}

void lcd_cmd(unsigned char command) {
   display_port = command;
   rs = 0;
   rw = 0;
   e = 1;
   msdelay(1);
   e = 0;
}



void clear_display_if_special() {
   if (C4 == 0) {  // Check if one of the special symbol keys is pressed
      lcd_cmd(0x80); // Clear screen
      msdelay(10);
   }
}
const unsigned char sine_table[50] = {
  128, 143, 159, 174, 189, 202, 215, 226,
235, 243, 249, 253, 255, 255, 253, 249,
243, 235, 226, 215, 202, 189, 174, 159,
143, 128, 112, 96, 81, 66, 53, 40,
29, 20, 12, 6, 2, 0, 0, 2,
6, 12, 20, 29, 40, 53, 66, 81,
96, 112
};
void generate_sine_wave(long int freq_sine) {

 /*//sine_flag =1;
   unsigned char index = 0;
   unsigned int period_us = 1000000 / freq_sine;
   unsigned int tmr_reload_value = 65536 - (unsigned int)((float)period_us * 11.0592 / 12.0);
 if (freq_sine <= 0) {
      // If frequency is not positive, exit the function
      return;
   }
 TMOD &= 0xF0;
   TMOD |= 0x01;
```

```
 TH0 = (unsigned char)(tmr_reload_value >> 8);
   TL0 = (unsigned char)tmr_reload_value;
   TR0 = 1;*/
 unsigned char index = 0; // Index for the sine lookup table
 unsigned char out_voltage=0;
   // Set up the output pin for the sine wave
   P2 = 0x00; // All pins are outputs

   // Set up Timer 0 for mode 1 (16-bit timer)
   TMOD = 0x01;

   // Set the timer reload value
   TH0 = TL0 = 65536 - (unsigned long int)((11059200UL / 12) * (1.0 / (4 * freq_sine)));
   while (1) {
      out_voltage = sine_table[index++];
      if (index >= 50) {
         index = 0;
      }
   P2 = out_voltage;
      nop();
   }
}


void generate_square_wave(long int freq_square) {
 unsigned int period_us;
   unsigned int tmr_reload_value;
 if (freq_square <= 0) {
      // If frequency is not positive, exit the function
      return;
   }
   // Calculate the period (in microseconds) for the desired frequency
   period_us = 1000000 / freq_square;
   // Calculate the timer reload value based on the system clock frequency (assuming 11.0592 MHz)
   tmr_reload_value = 65536 - (unsigned long int)((float)period_us * 11.0592 / 12.0);

   // Set up Timer 0 in Mode 1 (16-bit timer with auto-reload)
   TMOD &= 0xF0;  // Clear Timer 0 mode bits
   TMOD |= 0x01;  // Set Timer 0 to Mode 1
// Set the initial value for Timer 0
   TH0 = (unsigned char)(tmr_reload_value >> 8);  // Load high byte
   TL0 = (unsigned char)tmr_reload_value;        // Load low byte

   // Start Timer 0
   TR0 = 1;
```

```
 // Main loop
   while (1) {
      // Check if Timer 0 overflowed
      if (TF0) {
         // Toggle Port 2 for DAC (assuming all pins are configured as outputs)
         P2 = ~P2;
         // Clear Timer 0 overflow flag
         TF0 = 0;
         // Reload Timer 0 with initial value
         TH0 = (unsigned char)(tmr_reload_value >> 8);  // Load high byte
         TL0 = (unsigned char)tmr_reload_value;        // Load low byte
      }
   }
}

void generate_triangular_wave(long int freq_triangle) {
  unsigned int period_us;
    unsigned int tmr_reload_value;
    unsigned int count = 0; // initialize the counter to 0
    bit ascending = 1; // flag to indicate ascending or descending
  if (freq_triangle <= 0) {
      // If frequency is not positive, exit the function
      return;
    }
  // Set the desired frequency of the triangular wave
//#define FREQ 1000 // Hz

// Set the reference voltage for the triangular wave
#define VREF 5.0 // volts

// Calculate the step size
#define STEP_SIZE (255 / VREF)

// The timer reload value
//#define TMR_RELOAD (65536 - (11059200 / 12) * (1 / freq_triangle))
  // Calculate the period (in microseconds) for the desired frequency
    period_us = 1000000 / freq_triangle;
    // Calculate the timer reload value based on the system clock frequency (assuming 11.0592 MHz)
       tmr_reload_value = 65536 - (unsigned long int)((float)period_us * 11.0592 / 12.0)* (1 /
freq_triangle);
    // Set up the output pin for the triangular wave
    P2 = 0x00; // all pins are outputs
```

```
// Set up timer 0 for mode 1 (16-bit timer)
   TMOD = 0x01;

   // Set the timer reload value
   TH0 = (tmr_reload_value >> 8) & 0xFF;
   TL0 = tmr_reload_value & 0xFF;

   // Main loop
   while (1)
   {
      // Generate triangular waveform
      if (count == 0) {
         ascending = 1; // Start ascending
      } else if (count == 255) {
         ascending = 0; // Start descending
      }

      if (ascending) {
         count++;
      } else {
         count--;
      }

      // Output triangular wave value
      P2 = count;

      // Add a small delay to control the frequency
      // Adjust the delay as needed to achieve the desired frequency
      nop ();
   nop ();

   }
}



void generate_sawtooth_wave(long int freq_sawtooth) {
  unsigned int period_us;
    unsigned int tmr_reload_value;
  unsigned char count = 0; // initialize the counter to 0
  bit ascending = 1; // flag to indicate ascending or descending
  // Calculate the period (in microseconds) for the desired frequency
    period_us = 1000000 / freq_sawtooth;
    // Calculate the timer reload value based on the system clock frequency (assuming 11.0592 MHz)
        tmr_reload_value = 65536 - (unsigned long int)((float)period_us * 11.0592 / 12.0)*(1 /
freq_sawtooth);
```

```
if (freq_sawtooth <= 0) {
      // If frequency is not positive, exit the function
      return;
   }
  // Set the desired frequency of the triangular wave
//#define FREQ 1000 // Hz

// Set the reference voltage for the triangular wave
#define VREF 5.0 // volts

// Calculate the step size
#define STEP_SIZE (255 / VREF)

   P2 = 0x00; // all pins are outputs

   // Set up timer 0 for mode 1 (16-bit timer)
   TMOD = 0x01;

   // Set the timer reload value
   TH0 = (tmr_reload_value >> 8) & 0xFF;
   TL0 = tmr_reload_value & 0xFF;

   // Main loop
   while (freq_selected)
   {
      // Generate triangular waveform
      if (count == 0) {
         ascending = 1; // Start ascending
      } else if (count == 255) {
         ascending = 0; // Start descending
      }

      if (ascending) {
         count++;
      } else {
         count=0;
      }

      // Output triangular wave value
      P2 = count;

      // Add a small delay to control the frequency
      // Adjust the delay as needed to achieve the desired frequency
      nop ();
   }
}

}
```

```
// Function to handle 'e' press
void handle_e_press() {
   freq_int = atoi(freq_in); // Convert frequency input to integer
   freq_selected = 1; // Set flag indicating frequency is selected

        // Frequency is selected, wait for 'e' press to generate waveform
        if (sine_flag) {
           generate_sine_wave(freq_int);
        } else if (square_flag) {
           generate_square_wave(freq_int);
        } else if (triangle_flag) {
           generate_triangular_wave(freq_int);
        } else if (sawtooth_flag) {
           generate_sawtooth_wave(freq_int);
        } else {
           nop();
        }

}

void lcd_data(unsigned char disp_data) {
   display_port = disp_data;
   rs = 1;
   rw = 0;
   e = 1;
   msdelay(1);
   e = 0;
}

void lcd_init() {
   lcd_cmd(0x38);  // for using 2 lines and 5X7 matrix of LCD
   msdelay(10);
   lcd_cmd(0x0F);  // turn display ON, cursor blinking
   msdelay(10);
   lcd_cmd(0x01);  // clear screen
   msdelay(10);
   lcd_cmd(0x81);  // bring cursor to position 1 of line 1
   msdelay(10);
}
void row_finder1() {
   R1 = R2 = R3 = R4 = 1; // Set all rows to high
   C1 = 0;C2 = C3 = C4 = 1; // Ground column 1
  msdelay(20); // Wait for signals to settle
```

```
 if (R1 == 0) {
      //lcd_data('7');
      //freq_in[i++] = '7';
   lcd_data('1');
      freq_in[i++] = '1';
   }
   else if (R2 == 0) {
      lcd_data('4');
      freq_in[i++] = '4';
   }
   else if (R3 == 0) {
      //lcd_data('1');
      //freq_in[i++] = '1';
   lcd_data('7');
      freq_in[i++] = '7';
   }
   else if (R4 == 0) {

   atoi_test =0;
   //freq_int = atoi(freq_in); // Convert the string to an integer
   handle_e_press();
      lcd_data('e');
   }
}
void row_finder2() {
   R1 = R2 = R3 = R4 = 1;
   C2 = 0; C1 = C3 = C4 = 1;
 msdelay(20); // Wait for signals to settle
   if (R1 == 0) {
      //lcd_data('8');
      //freq_in[i++] = '8';
   lcd_data('2');
      freq_in[i++] = '2';
   }
   if (R2 == 0) {
      lcd_data('5');
      freq_in[i++] = '5';
   }
   if (R3 == 0) {
   lcd_data('8');
      freq_in[i++] = '8';
      //lcd_data('2');
      //freq_in[i++] = '2';
   }
   if (R4 == 0) {
      lcd_data('0');
      freq_in[i++] = '0';
   }
}
```

```
void row_finder3() {
   R1 = R2 = R3 = R4 = 1;
   C1 = C2 = C4 = 1;
   C3 = 0;  // Activate column 3
   msdelay(20);  // Wait for signals to settle

   if (R1 == 0) {
      lcd_data('3');
      freq_in[i++] = '3';
   }
   else if (R2 == 0) {
      lcd_data('6');
      freq_in[i++] = '6';
   }
   else if (R3 == 0) {
      lcd_data('9');
      freq_in[i++] = '9';
   }
   else if (R4 == 0) {
      msdelay(5);  // Adjust delay as needed
      lcd_cmd(0xC0); // Move cursor to the beginning of the second line
      msdelay(10);
      lcd_data('F');
      msdelay(15);
      lcd_data('r');
      msdelay(15);
      lcd_data('e');
      msdelay(15);
      lcd_data('q');
      lcd_data(' ');
      msdelay(15);
      lcd_data('{');
      msdelay(15);
      lcd_data('H');
      lcd_data('z');
      msdelay(15);
      lcd_data('}');
      lcd_data(' ');
   }
}

void row_finder4() {
   R1 = R2 = R3 = R4 = 1;
   C1 = C2 = C3 = 1;
   C4 = 0; // Activate column 4
   msdelay(10); // Wait for signals to settle
```

```
if (R1 == 0) {
      clear_display_if_special(); // Clear the display if a special symbol key is pressed
      lcd_data('S');
      msdelay(15);
      lcd_data('i');
      msdelay(15);
      lcd_data('n');
      msdelay(15);
      lcd_data('e');
      //generate_sine_wave(freq_int);
   sine_flag =1;
      lcd_data(' ');
      lcd_data(' ');
      lcd_data(' ');
      lcd_data(' ');
      lcd_data(' ');
      lcd_data(' ');

   }
   else if (R2 == 0) {
      clear_display_if_special(); // Clear the display if a special symbol key is pressed
      lcd_data('S');
      msdelay(15);
      lcd_data('q');
      msdelay(15);
      lcd_data('u');
      msdelay(15);
      lcd_data('a');
      msdelay(15);
      lcd_data('r');
      msdelay(15);
      lcd_data('e');
   square_flag =1;
      //generate_square_wave(freq_int); // Generate square wave continuously
      lcd_data(' ');
      lcd_data(' ');
      lcd_data(' ');
      lcd_data(' ');
      lcd_data(' ');
   }
else if (R3 == 0) {
      clear_display_if_special(); // Clear the display if a special symbol key is pressed
      lcd_data('T');
      msdelay(15);
      lcd_data('r');
      msdelay(15);
      lcd_data('i');
      msdelay(15);
```

```
lcd_data('a');
 msdelay(15);
 lcd_data('n');
 msdelay(15);
 lcd_data('g');
 msdelay(15);
 lcd_data('l');
 msdelay(15);
 lcd_data('e');
 //generate_triangular_wave(freq_int);
 triangle_flag =1;
 lcd_data(' ');
 lcd_data(' ');
 lcd_data(' ');
 }
else if (R4 == 0) {
      clear_display_if_special(); // Clear the display if a special symbol key is pressed
      lcd_data('S');
      msdelay(15);
      lcd_data('a');
      msdelay(15);
      lcd_data('w');
      msdelay(15);
      lcd_data('t');
      msdelay(15);
      lcd_data('o');
      msdelay(15);
      lcd_data('o');
      msdelay(15);
      lcd_data('t');
      msdelay(15);
      lcd_data('h');
   //generate_sawtooth_wave(freq_int);
   sawtooth_flag =1;
      lcd_data(' ');
      lcd_data(' ');
      lcd_data(' ');
   }
 }
void scan_keypad() {
   // Ground all rows
   R1 = R2 = R3 = R4 = 0;
   // Set all columns as input
   C1 = C2 = C3 = C4 = 1;
 msdelay(15);
   if(C1==0)
   row_finder1();
   else if(C2==0)
      row_finder2();
   else if(C3==0)
      row_finder3();
   else if(C4==0)
      row_finder4();
}
```

```
void display_function_generator() {
    lcd_cmd(0x01); // Clear screen
    msdelay(10);

    lcd_cmd(0x80); // Set cursor to the beginning of the first line
    msdelay(10);
    lcd_data('F');
    msdelay(15);
    lcd_data('U');
    msdelay(15);
    lcd_data('N');
    msdelay(15);
    lcd_data('C');
    msdelay(15);
    lcd_data('T');
    msdelay(15);
    lcd_data('I');
    msdelay(15);
    lcd_data('O');
    msdelay(15);
    lcd_data('N');
    msdelay(15);

    lcd_cmd(0xC0); // Set cursor to the beginning of the second line
    msdelay(10);
    lcd_data('G');
    msdelay(15);
    lcd_data('E');
    msdelay(15);
    lcd_data('N');
    msdelay(15);
    lcd_data('E');
    msdelay(15);
    lcd_data('R');
    msdelay(15);
    lcd_data('A');
    msdelay(15);
    lcd_data('T');
    msdelay(15);
    lcd_data('O');
    msdelay(15);
    lcd_data('R');
    msdelay(15);
}

void clear_display() {
    lcd_cmd(0x01); // Clear screen
    msdelay(10);
}
```

```
void main() {
   unsigned char keypad_pressed = 0;

   // Initialize LCD
   lcd_init();

   // Display "FUNCTION GENERATOR" on reset
   display_function_generator();

   while (1) {
      // Check if any keypad button is pressed
      scan_keypad();
      if (keypad_pressed) {
         // Clear the display
         clear_display();
         // Reset the flag to indicate no button is pressed
         keypad_pressed = 0;
      }
   }
}
```

Kevin Andani
21BEC008

Vedanti Patel
21BEC095

Het Patel
22BEC050

Kaival Prajapati
22BEC095