Arduino based Function Generator

Sheikh Zeeshan Basar (49)

Department of Electrical and Electronics Engineering; Manipal Institute of Technology, Manipal

Abstract:

IN any electronics laboratory or the workbench of an electronics hobbyist, a function generator is an essential piece of equipment. As a part of MI Lab mini project, an Arduino based Function Generator is developed. It can generate square and sinusoidal waveform from frequency 1Hz to 10kHz. In this project, Arduino Nano is used. The frequency of the waveform is adjusted using a rotary encoder.

Introduction:

A function generator is one of the most useful pieces of equipment in an electronics laboratory or a hobbyist's workbench. It is used to generate various forms of test signals like a square, sinusoidal, and triangular wave to name a few. Some applications of function generator include RF signal generations, digital pattern generation, pitch generations, and arbitrary waveform generation.

A simple function generator, like the one developed here, is a device which can produce simple repetitive waveforms. The function generator developed here uses the Tone and TimerOne libraries of Arduino to generate square and sinusoidal waveforms. The library Rotary is used to fetch and convert the rotations of the encoder to a number and set the frequency.

Objective

1. **To generate square and sinusoidal waveforms.**
2. **To vary the frequency of said waveforms via an external input.**

Methodology

There were four steps involved in developing this function generator. First, was the generation of the square waveform. Second, was to generate a sinusoidal waveform. Third, was to integrate the rotary encoder and convert the rotations and switch clicks to meaningful data. Lastly, to vary the frequency of the said waveforms.

1. SQUARE WAVEFORM

   The square waveform was generated using the Tone library of Arduino. This library uses the hardware timers on the microcontroller to generate square-wave tones. The functions tone(P, F) or tone(P, F, D) calls this library. Where P is the pin on which to generate the tone, the F is the frequency of the sound in hertz(Hz), and D is the duration of the sound in milliseconds(optional).

   This setup generates the square wave on digital pin 9, and the rotary encoder sets the variable frequency.

2. SINUSOIDAL WAVEFORM

   The sinusoidal waveform was generated using the SPWM method. In this method, a frequency is set as a constant in the code, and the variable duty is set using a recursive difference equation. The TimerOne library contains all the functions necessary to generate sinusoidal waveform. This function generator uses a simple, single-pole RC LPF to convert SPWM to a sinusoidal waveform. This setup uses a 5.6kΩ(±5%) resistor and a 10nF ceramic capacitor.

3. ROTARY ENCODER

   A typical rotary encoder emits a two-bit gray code from its three pins. Every step in the output (often accompanied by a physical 'click') generates a specific sequence of output codes on its pins.

The sequence of codes on o/p pins when moving from one step to next

Position	Byte 1	Byte 2
Step 1	0	0
¼	1	0
½	1	1
¾	0	1
Step 2	0	0

To decode the sequence of this code, that is to detect the direction, the library uses a state machine which returns DIR_NONE, DIR_CW, and DIR_CCW depending on the direction of rotation. This library includes software debouncing as well.

Variables **freq** and **multiplier** were globally initialized to 100 and 0.1 respectively. The variable **freq** increments by the value of **multiplier** if the encoder rotates in clockwise direction, and decrements if rotated counter-clockwise.

The encoder switch is a normally closed switch, so whenever it is pressed a HIGH-LOW-HIGH sequence occurs. The digitalRead() function of Arduino detects this LOW and is used to detect the switch state change. The **multiplier** increases ten-fold on every encoder switch press. The maximum value for the **multiplier** is set as 1000.

4. VARIABLE FREQUENCY

   The variable **freq** sets the frequency of the waveforms. The function tone() directly takes this variable directly as the second parameter. Since the frequency of the sinusoidal waveform is a global variable, it cannot be set via an external input. The only way to adjust the sinusoidal waveform is by hardcoding the value.

Result Analysis:

This setup was able to generate a square and sinusoidal waveforms of variable frequency. The sinusoidal waveform has minimal harmonic distortion which was confirmed by the FFT functionality of the oscilloscope.

Conclusion

The function generator developed here was able to generate a square and sinusoidal waveforms with variable frequency. The sinusoidal waveform has minimal harmonic content as confirmed visually using the FFT functionality of the oscilloscope.

This system can be further developed to include the following in the order of priority:

1. Externally variable frequency for the sinusoidal waveform.
2. LCD screen to display the frequency of the waveform. Currently, the Serial Monitor of Arduino IDE displays the frequency.
3. Triangular and sawtooth waveforms.
4. Single channel output. Currently, separate pins generate sinusoidal and square waveforms.

References

1. Rotary library by **brainlow** on Github, https://github.com/brianlow/Rotary
2. TimerOne library by PaulStoffregen on Github, https://github.com/PaulStoffregen/TimerOne
3. General Arduino debugging and recursive equation for SPWM duty from Arduino forum, https://forum.arduino.cc