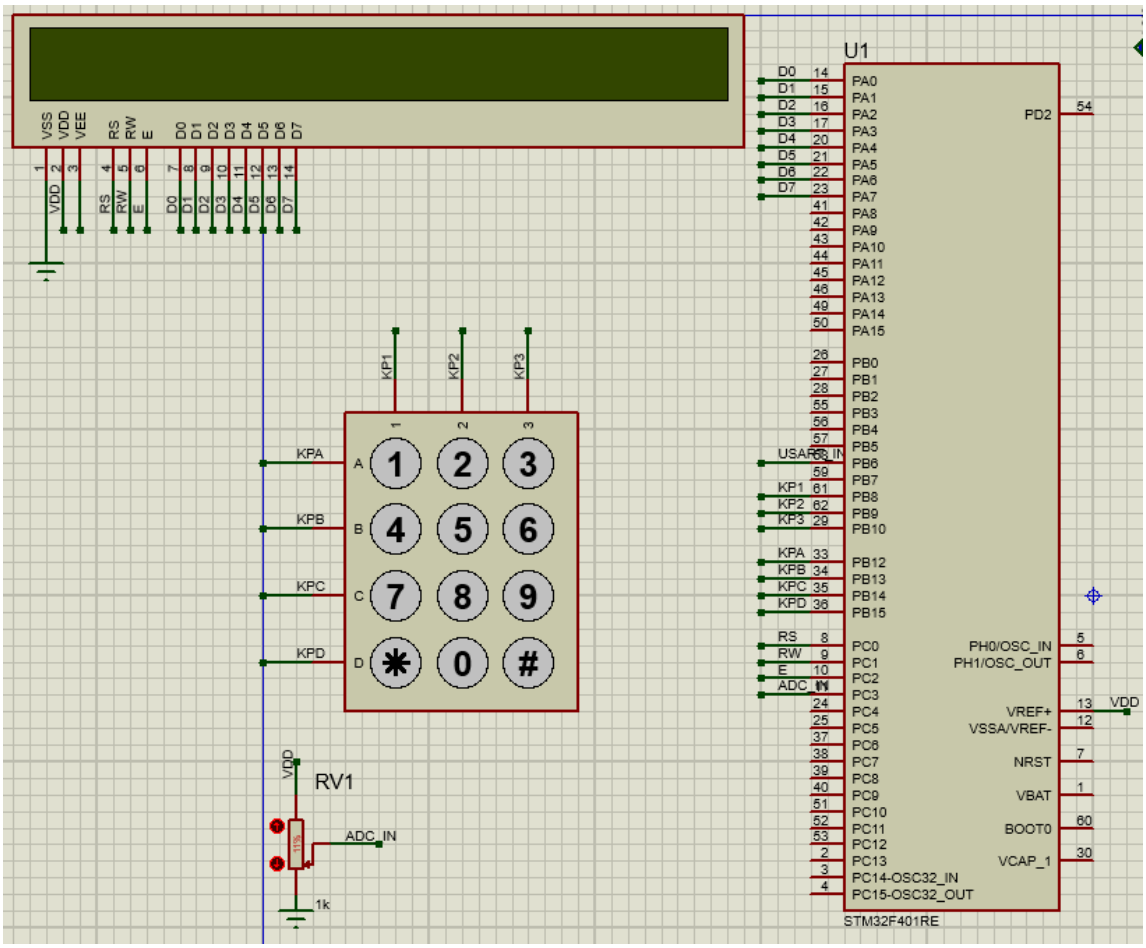
به نام خدا

نیکی نظران

۹۵۲۴۳۰۶۷

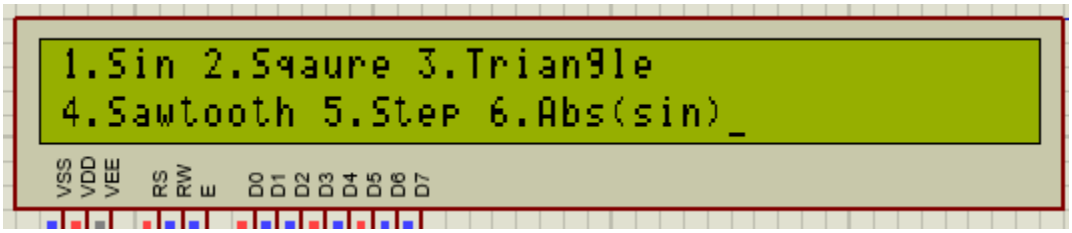
**STM1: Sender**

در ابتدای کد این میکروکنترولر، سه پورت PA و PB و PC آماده‌سازی شدند. پورت A برای نمایش مقادیر و پیام‌ها روی LCD استفاده می‌شود. پین PBA6 برای فرستادن سریال اطلاعات به میکروکنترولر دیگر و از PBA8 تا PBA15 برای کنترل keypad استفاده می‌شود. سه پین اول PC برای کنترل LCD و پین چهارم آن برای خواندن مقادیر آنالوگ از مقاومت متغیر است.




سپس LCD آماده‌سازی می‌شود و شماره‌ی دانشجویی برای ۱ ثانیه نمایش داده می‌شود. (البته این زمان بسته به تند یا کند بودن پردازنده متغیر است.)

در مرحله‌ی بعد گزینه‌ها برای انواع شکل موج‌ها نشان داده می‌شود و میکروکنترولر منتظر میماند تا کاربر عددی از ۱ تا ۵ وارد کند.




حال کاربر باید با تغییر دادن مقاومت متغیر مقداری برای زمان اعمال موج و سپس فرکانس انتخاب کند. در هر دو مرحله با تغییر مقدار مقاومت، زمان یا فرکانس متناظر روی صفحه نمایش داده می‌شود.

پس از ثبت زمان و فرکانس مورد نظر، اطلاعات در آرایه‌ی figs در ۴ char ذخیر می‌شوند و به STM2 فرستاده می‌شود. برای ذخیره‌ی اطلاعات دقیقا ۲۷ بیت نیاز داریم: ۳ بیت برای انتخاب شکل موج، ۱۴ بیت برای زمان اعمال، ۱۰ بیت برای فرکانس.
نحوه‌ی ذخیره‌ی مقادیر در figs به این صورت است(۴ بایت تولید شده به منظور دیباگ روی LCD نمایش داده می‌شود):

```
figs[0] = actualTime << 3;
figs[0] |= chosenWave;
figs[1] = actualTime >> 5;
figs[2] = actualTime >> 13;
figs[2] |= actualFreq << 1;
figs[3] = actualFreq >> 7;
```

سپس USART آماده‌سازی می‌شود. در تابع initUSART ابتدا کلاک برای GPIOB و USART1 روشن می‌شود و alternate function برای پین ۶ تنظیم می‌شود. این پین باید به پین ۷ میکروکنترولر دیگر متصل شود. فرکانس مشخص می‌شود. سپس USART1 را برای تنظیمات disable میکنیم. تعداد بیت‌های داده را ۸ بیت با ۱ بیت stop و بدون هرگونه interrupt تنظیم میکنیم و در انتها USART1 را دوباره enable می‌کنیم. برا ی نوشتن به صورت سریال از تابع writeUSART استفاده می‌شود که ابتدا چک می‌کند که آیا بافر خالی است که بتواند داده‌ی جدید بفرستد یا خیر. در صورت خالی بودن داده را روی رجیستر DR می‌گذارد تا فرستاده شود.

ابتدا ۱ بایت ۰ برای STM2 فرستاده می‌شود که مشخص کند که کار STM2 باید شروع شود. عمل فرستادن داده را برای هر ۴ بایت figs انجام می‌دهیم.

**STM2: receiver**

در این میکروکنترولر ابتدا پورت‌های PA و PB آماده‌سازی می‌شوند. پورت PA برای فرستاده داده به DAC و پین PB7 برای گرفتن داده‌ی سریال استفاده می‌شود.

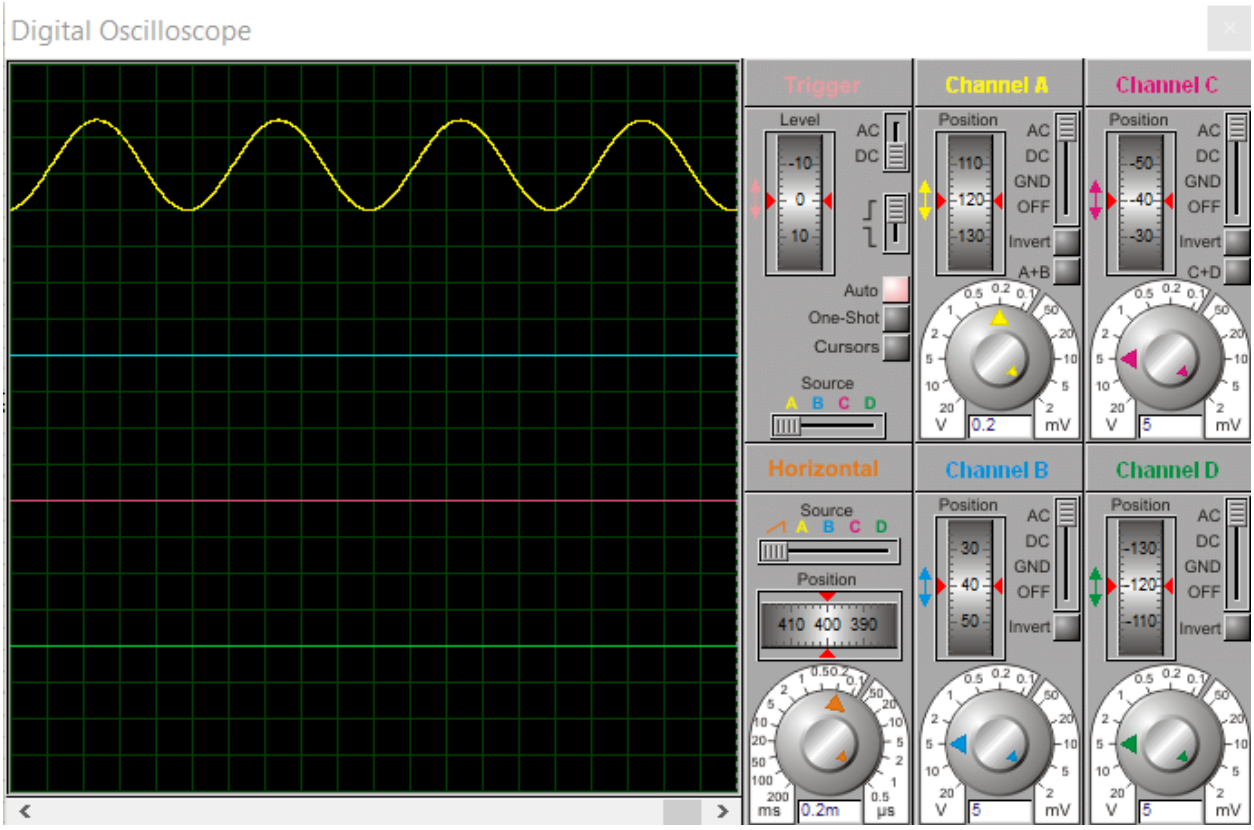
تابع initUSART برای این میکروکنترولر بسیار شبیه به قبلی است با این تفاوت که برای پین ۷ انجام می‌شود.

این میکروکنترولر منتظر میماند تا بایت ۰ نشانگر شروع عملیاتش را دریافت کند و سپس ۴ بایت figs را دریافت می‌کند و مقادیر استخراج شده از آن را در ۳ متغیر chosenWave و time و freq ذخیره می‌کند. با استفاده از این مقادیر، دوره تناوب یعنی t (به میلی‌ثانیه) و count یعنی تعداد دوره تناوب‌های خواسته شده را محاسبه می‌کند.

```
chosenWave = figs[0] & 0x07;
time = ((figs[2] & 0x01) << 13) | (figs[1] << 5) | (figs[0] >> 3);
freq = (figs[3]) | (figs[2] >> 1);
t = 1000 / freq;
count = time * freq;
```

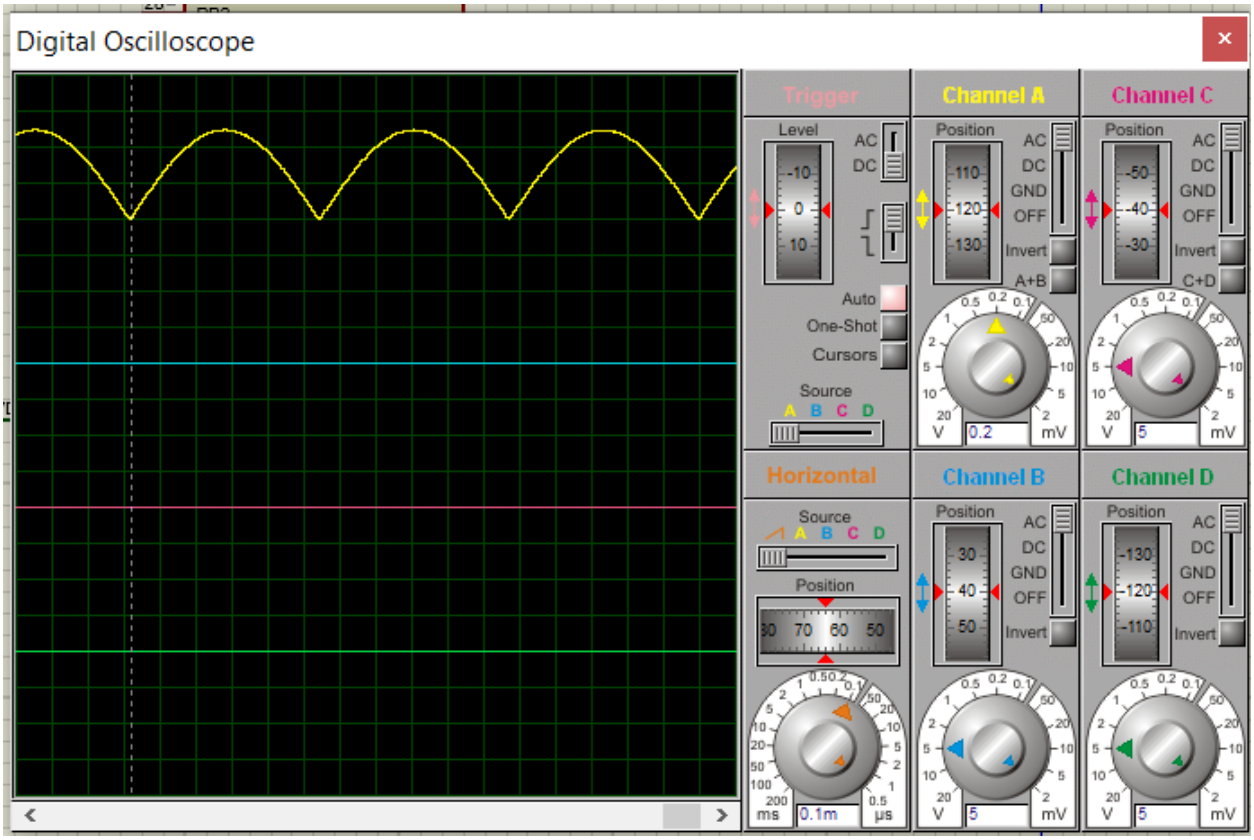
**شکل موج‌ها:**

سینوسی:



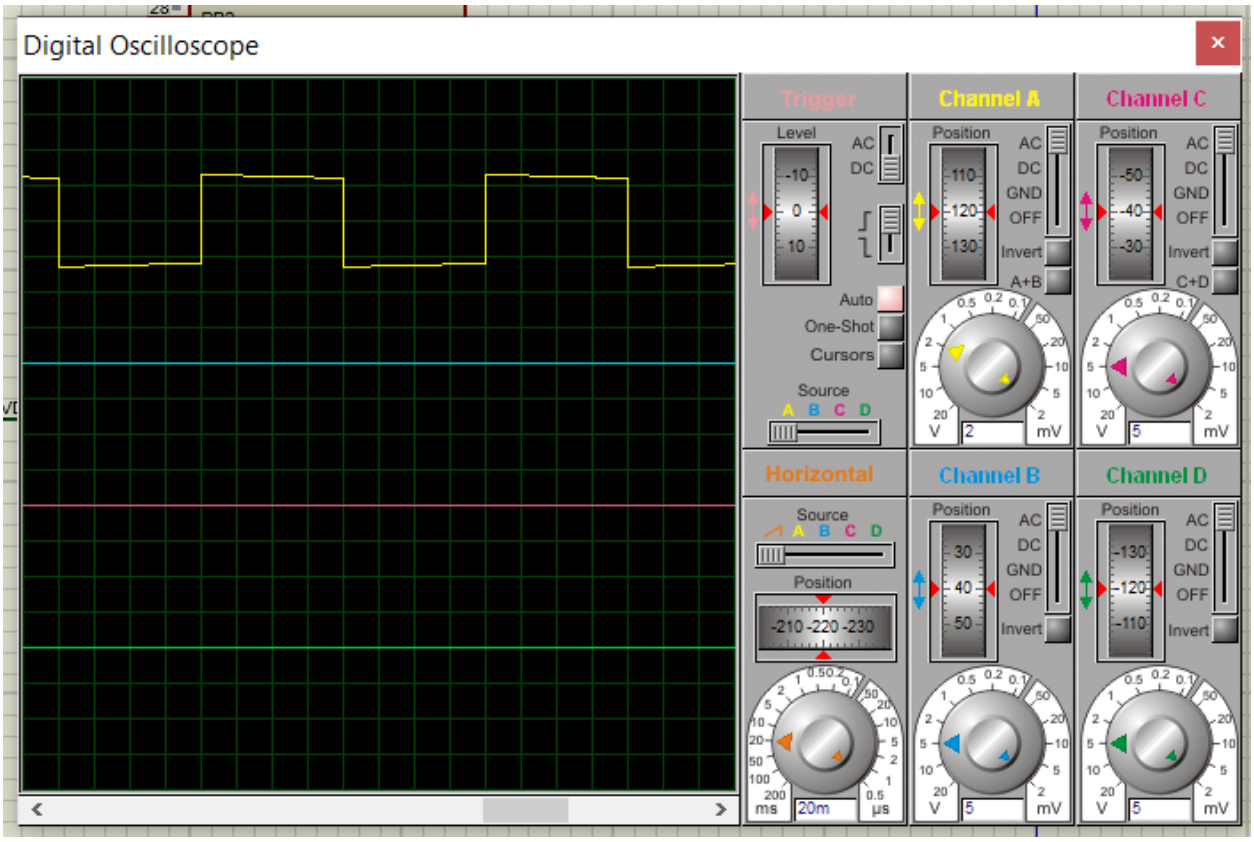

روش ایجاد این شکل موج به این صورت است که آرایه‌ای به نام sins با ۴۰۰ مقدار سینوس از زاویه‌ی ۰ تا ۳۶۰ داریم، هر t/100 میلی ثانیه یکی از این مقادیر را نشان می‌دهیم.

قدر مطلق سینوسی:

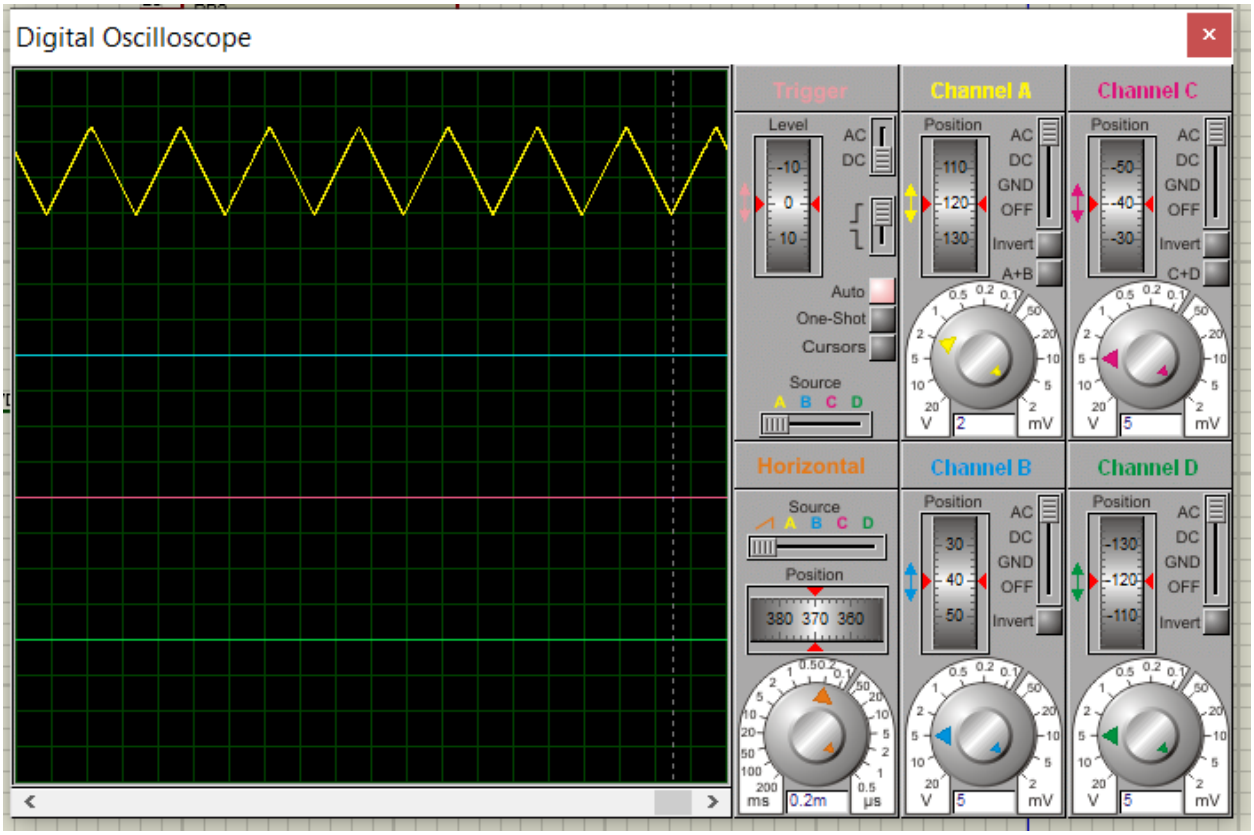



در این روش هم مثل سینوسی عمل میکنیم اما به جای زوایای ۰ تا ۳۶۰، از زاویه‌ی ۰ تا ۱۸۰ استفاده می‌کنیم.

مربعی:

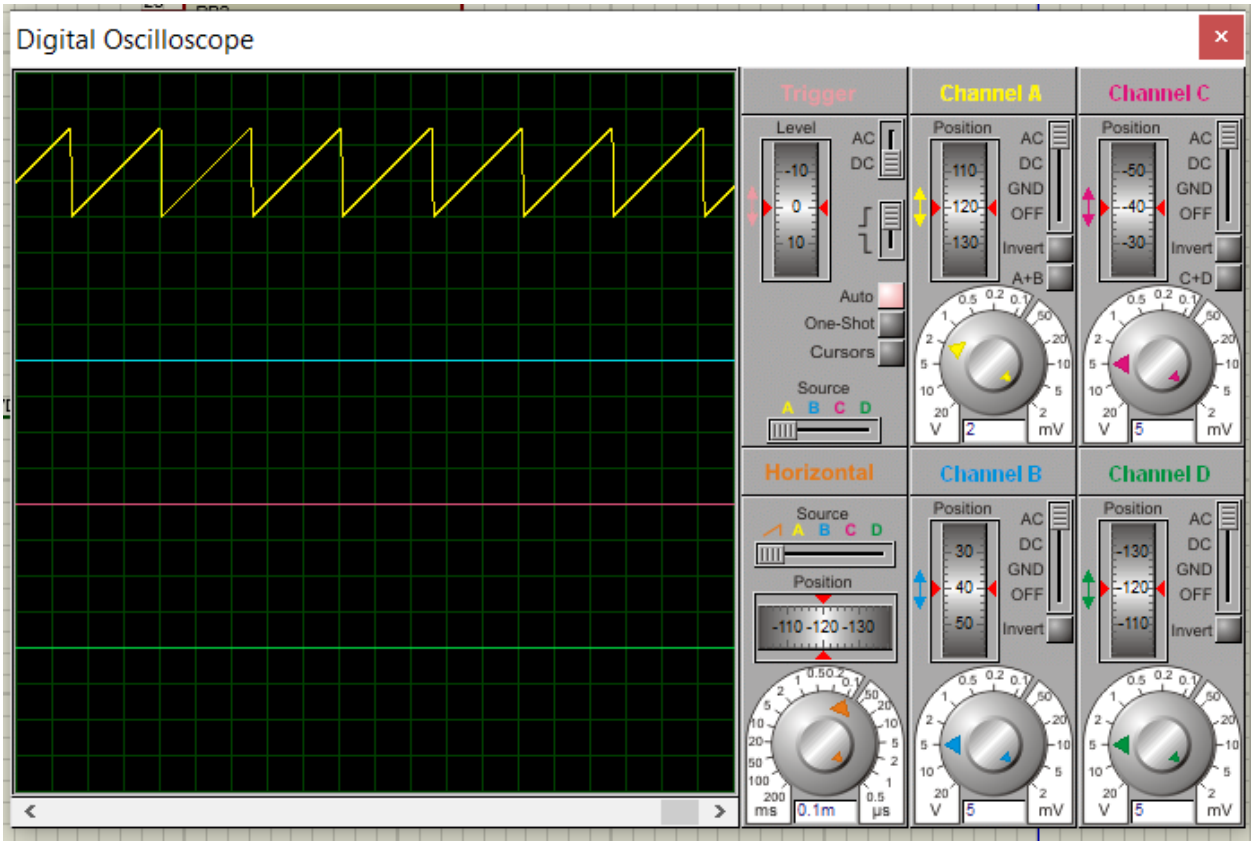

Digital Oscilloscope
Trigger
Level
AC
DC
Auto
One-Shot
Cursors
Source
A B C D
Channel A
Position
AC
DC
GND
OFF
Invert
A+B
Channel C
C+D
Horizontal
20m
Channel B
Channel D
mV
V
ms
µs

مثلثی:



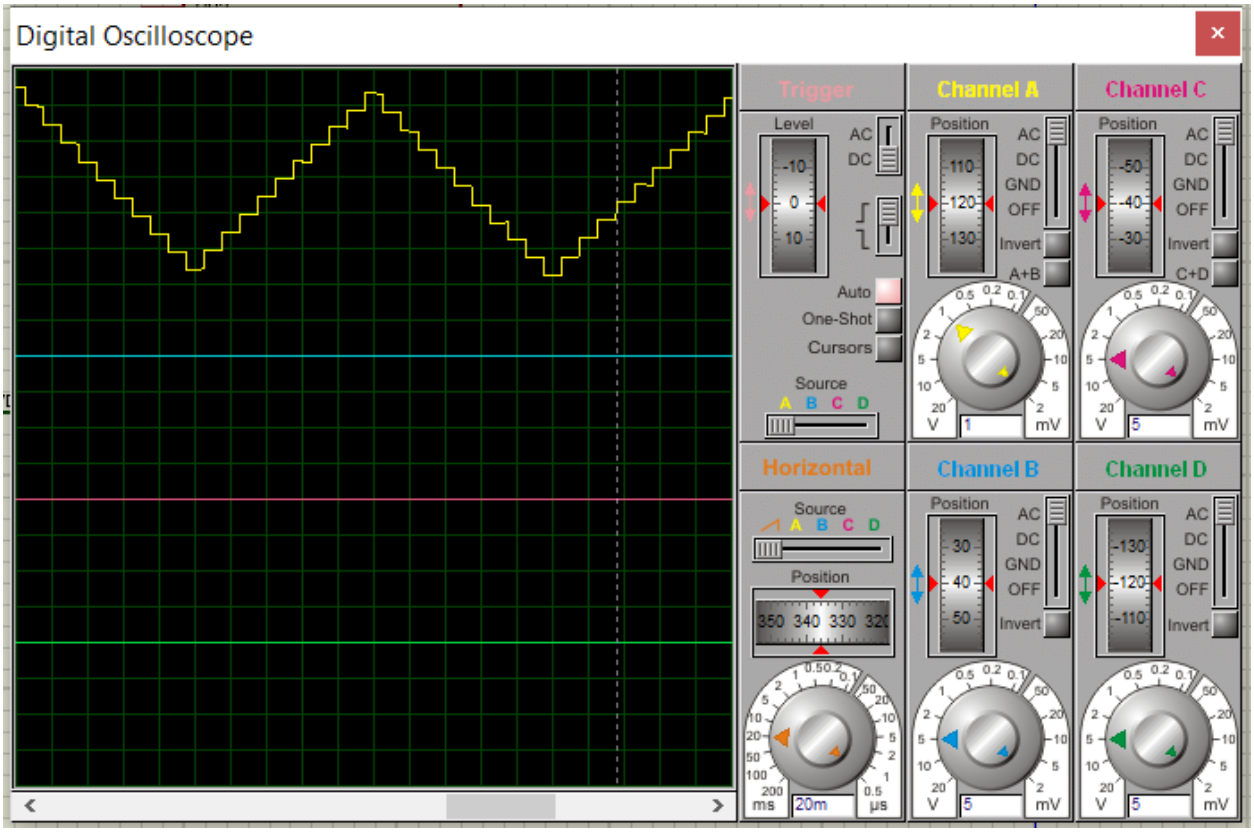

برای این شکل موج نیز ۱۰۰ مقدار از قبل در آرایه‌ی triangleValues ذخیره شده است و از آنها با تاخیرهای مختلف استفاده می‌شود.

دندان‌اره‌ای:




برای این شکل موج نیز ۱۰۰ مقدار از قبل در آرایه‌ی triangleValues ذخیره شده است و از آنها با تاخیرهای مختلف استفاده می‌شود. فرق این موج با مثلثی این است که در این موج قسمت پایین رونده وجود ندارد.

پله‌ای:




برای این شکل موج ۲۰ مقدار از قبل در آرایه‌ی stepValues ذخیره شده است و از آنها با تاخیرهای مختلف استفاده می‌شود.