15 July 2022

Software project of Electrical circuit course
Professor Hadi

Mohammad Parsa Dini
AmirHossein Naghdi
Ali Sadeghian
Amirreza Tanevardi

1 Introduction

As we learned in electrical circuit course, the main purpose of the course is designing a circuit with a special output which we want.
The most important part of this designing is to generate proper signal to check the output will be same with what we want or not.
So to reach this we need to have some devices to generate and measure which is known as Function generator and oscilloscope in Electrical circuit lab.
So in this project we will design a program which will generate signals by using digital generator methods and also an oscilloscope to find these signals kind.

2 Research

1.Function Generator:
A function generator is usually a piece of electronic test equipment or software used to generate different types of electrical waveforms over a wide range of frequencies. Some of the most common waveforms produced by the function generator are the sine wave, square wave, triangular wave and sawtooth shapes. These waveforms can be either repetitive or single-shot (which requires an internal or external trigger source).

Figure 1: An analog function generator 1990.

2.Oscilloscope:
An oscilloscope (informally a scope) is a type of electronic test instrument that graphically displays varying electrical voltages as a two-dimensional plot of one or more signals as a function of time. The main purposes are to display repetitive or single waveforms on the screen that would otherwise occur too briefly to be perceived by the human eye. The displayed waveform can then be analyzed for properties such as amplitude, frequency, rise time, time interval, distortion, and others. Originally, calculation of these values required manually measuring the waveform against the scales built into the screen of the instrument.[1] Modern digital instruments may calculate and display these properties directly.

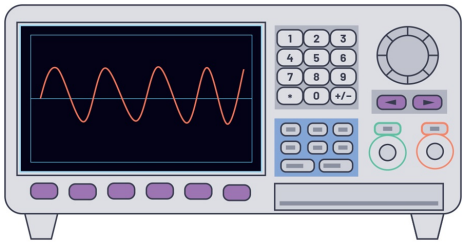


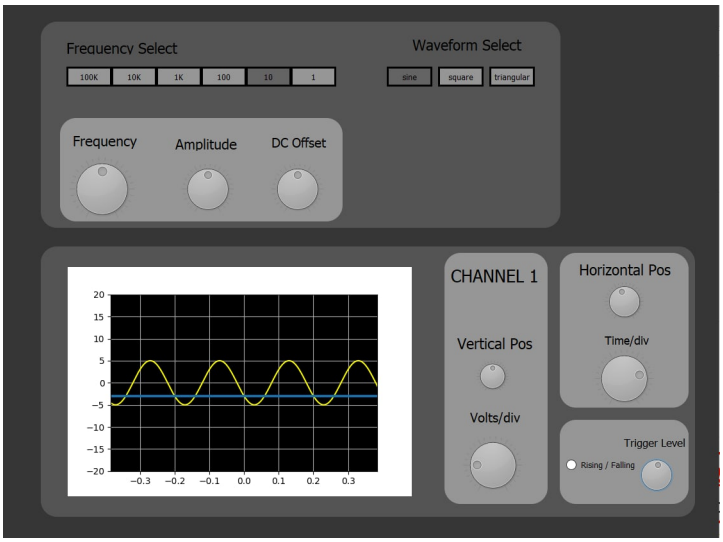
Figure 2: oscilloscope showing a trace with standard inputs and controls.

3 applications

We had two choices to do this project , both MATLAB and python programming languages was offered by Dr.Hadi to us . But we decide to do this by python because it was much better than MATLAB in graphics performance and also it is a language with object oriented structure which is learned by all members of group.

4 GUI

There is a picture of our generator and oscilloscope which is made by python PyQt5 library.



5 Code description

5.1 Gui of welcome page

By using qtdesigner (special app to design graphical pages for pyqt5 library) we design this page and also the gui of devices page.

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'welcome-page-gui.ui'
#
# Created by: PyQt5 UI code generator 5.15.4
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(1118, 834)
        Dialog.setStyleSheet("background-color: rgb(54, 54, 54);")
        self.label = QtWidgets.QLabel(Dialog)
        self.label.setGeometry(QtCore.QRect(30, 40, 821, 151))
        self.label.setStyleSheet("font: 50pt \"MS Shell Dlg 2\";\n"
"background-color: rgb(54, 54,54);")
        self.label.setObjectName("label")
        self.start_button = QtWidgets.QPushButton(Dialog)
        self.start_button.setGeometry(QtCore.QRect(950, 750, 93, 28))
        self.start_button.setStyleSheet("font: 10pt \"MS Shell Dlg 2\";\n"
"background-color: rgb(100, 100, 100);")
        self.start_button.setObjectName("start_button")
        self.label_2 = QtWidgets.QLabel(Dialog)
        self.label_2.setGeometry(QtCore.QRect(30, 160, 911, 151))
        self.label_2.setStyleSheet("font: 50pt \"MS Shell Dlg 2\";\n"
"background-color: rgb(54, 54, 54);")
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(Dialog)
        self.label_3.setGeometry(QtCore.QRect(30, 300, 281, 101))
        self.label_3.setStyleSheet("font: 30pt \"MS Shell Dlg 2\";\n"
"background-color: rgb(54, 54,54);")
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(Dialog)
        self.label_4.setGeometry(QtCore.QRect(30, 390, 381, 71))
        self.label_4.setStyleSheet("font: 26pt \"MS Shell Dlg 2\";\n"
"background-color: rgb(54, 54,54);")
```

```
        self.label_4.setObjectName("label_4")
        self.label_5 = QtWidgets.QLabel(Dialog)
        self.label_5.setGeometry(QtCore.QRect(30, 480, 191, 51))
        self.label_5.setStyleSheet("font: 24pt \"MS Shell Dlg 2\";\
    n"
"background-color: rgb(54, 54,54);")
        self.label_5.setObjectName("label_5")
        self.label_6 = QtWidgets.QLabel(Dialog)
        self.label_6.setGeometry(QtCore.QRect(30, 540, 331, 31))
        self.label_6.setStyleSheet("font: 20pt \"MS Shell Dlg 2\";\
    n"
"background-color: rgb(54, 54,54);")
        self.label_6.setObjectName("label_6")
        self.label_7 = QtWidgets.QLabel(Dialog)
        self.label_7.setGeometry(QtCore.QRect(30, 580, 311, 41))
        self.label_7.setStyleSheet("font: 20pt \"MS Shell Dlg 2\";\
    n"
"background-color: rgb(54, 54,54);")
        self.label_7.setObjectName("label_7")
        self.label_8 = QtWidgets.QLabel(Dialog)
        self.label_8.setGeometry(QtCore.QRect(30, 630, 231, 41))
        self.label_8.setStyleSheet("font: 20pt \"MS Shell Dlg 2\";\
    n"
"background-color: rgb(54, 54,54);")
        self.label_8.setObjectName("label_8")
        self.label_9 = QtWidgets.QLabel(Dialog)
        self.label_9.setGeometry(QtCore.QRect(30, 680, 321, 31))
        self.label_9.setStyleSheet("font: 20pt \"MS Shell Dlg 2\";\
    n"
"background-color: rgb(54, 54,54);")
        self.label_9.setObjectName("label_9")
        self.label_10 = QtWidgets.QLabel(Dialog)
        self.label_10.setGeometry(QtCore.QRect(740, 30, 311, 321))
        self.label_10.setStyleSheet("background-color: rgb(54,
    54,54);")
        self.label_10.setText("")
        self.label_10.setPixmap(QtGui.QPixmap("sharif-university-
    logo-1-600x593.png"))
        self.label_10.setObjectName("label_10")

        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "ECSP"))
        self.label.setText(_translate("Dialog", "Electric Circuits"
    ))
        self.start_button.setText(_translate("Dialog", "Start"))
        self.label_2.setText(_translate("Dialog", "Software Project
    "))
        self.label_3.setText(_translate("Dialog", "Spring 2022"))
        self.label_4.setText(_translate("Dialog", "Professor : Dr.
    Hadi"))
        self.label_5.setText(_translate("Dialog", "Made By :"))
        self.label_6.setText(_translate("Dialog", "Mohammad Parsa
    Dini"))
```

```
        self.label_7.setText(_translate("Dialog", "Amir Hossein Naghdi"))
        self.label_8.setText(_translate("Dialog", "Ali Sadeghian"))
        self.label_9.setText(_translate("Dialog", "Amir Reza Tanevardi"))


#if __name__ == "__main__":
    #import sys
    #app = QtWidgets.QApplication(sys.argv)
    #Dialog = QtWidgets.QDialog()
    #ui = Ui_Dialog()
    #ui.setupUi(Dialog)
    #Dialog.show()
    # sys.exit(app.exec_())
```

5.2 Gui of devices page

we have control both oscilloscope and function generator in only one program and also make the graphical part by another python file the codes are like this: For GUI we have:

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'App.ui'
#
# Created by: PyQt5 UI code generator 5.15.4
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(1118, 834)
        Dialog.setStyleSheet("background-color: rgb(54, 54, 54);\n"
"border-color: rgb(255, 255, 255);")
        self.frame = QtWidgets.QFrame(Dialog)
        self.frame.setGeometry(QtCore.QRect(60, 30, 811, 321))
        self.frame.setStyleSheet("border-color: rgb(85, 170, 255);\n"
"background-color: rgb(84, 84, 84);\n"
"border-radius: 25px;")
        self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame.setObjectName("frame")
        self.pushButton_100k = QtWidgets.QPushButton(self.frame)
        self.pushButton_100k.setGeometry(QtCore.QRect(40, 70, 71, 31))
```

```
30          self.pushButton_100k.setStyleSheet("background-color: rgb
    (150, 150, 150);\n"
31 "border :3px solid black")
32          self.pushButton_100k.setCheckable(True)
33          self.pushButton_100k.setObjectName("pushButton_100k")
34          self.pushButton_100 = QtWidgets.QPushButton(self.frame)
35          self.pushButton_100.setGeometry(QtCore.QRect(250, 70, 71,
    31))
36          self.pushButton_100.setStyleSheet("background-color: rgb
    (150, 150, 150);\n"
37 "border :3px solid black")
38          self.pushButton_100.setCheckable(True)
39          self.pushButton_100.setObjectName("pushButton_100")
40          self.pushButton_1k = QtWidgets.QPushButton(self.frame)
41          self.pushButton_1k.setGeometry(QtCore.QRect(180, 70, 71,
    31))
42          self.pushButton_1k.setStyleSheet("background-color: rgb
    (150, 150, 150);\n"
43 "border :3px solid black")
44          self.pushButton_1k.setCheckable(True)
45          self.pushButton_1k.setObjectName("pushButton_1k")
46          self.pushButton_10k = QtWidgets.QPushButton(self.frame)
47          self.pushButton_10k.setGeometry(QtCore.QRect(110, 70, 71,
    31))
48          self.pushButton_10k.setStyleSheet("background-color: rgb
    (150, 150, 150);\n"
49 "border :3px solid black")
50          self.pushButton_10k.setCheckable(True)
51          self.pushButton_10k.setObjectName("pushButton_10k")
52          self.pushButton_10 = QtWidgets.QPushButton(self.frame)
53          self.pushButton_10.setGeometry(QtCore.QRect(320, 70, 71,
    31))
54          self.pushButton_10.setStyleSheet("background-color: rgb
    (150, 150, 150);\n"
55 "border :3px solid black")
56          self.pushButton_10.setCheckable(True)
57          self.pushButton_10.setObjectName("pushButton_10")
58          self.pushButton_1 = QtWidgets.QPushButton(self.frame)
59          self.pushButton_1.setGeometry(QtCore.QRect(390, 70, 71, 31)
    )
60          self.pushButton_1.setStyleSheet("border-color: rgb(0, 0, 0)
    ;\n"
61 "background-color: rgb(150,150,150);\n"
62 "border :3px solid black")
63          self.pushButton_1.setCheckable(True)
64          self.pushButton_1.setObjectName("pushButton_1")
65          self.pushButton_square = QtWidgets.QPushButton(self.frame)
66          self.pushButton_square.setGeometry(QtCore.QRect(620, 70,
    71, 47))
67          self.pushButton_square.setStyleSheet("background-color: rgb
    (150, 150, 150);\n"
68 "border :3px solid black")
69          self.pushButton_square.setText("")
70          icon = QtGui.QIcon()
71          icon.addPixmap(QtGui.QPixmap("square-wave (1).png"), QtGui.
    QIcon.Normal, QtGui.QIcon.Off)
72          self.pushButton_square.setIcon(icon)
```

```
        self.pushButton_square.setIconSize(QtCore.QSize(40, 30))
        self.pushButton_square.setCheckable(True)
        self.pushButton_square.setObjectName("pushButton_square")
        self.pushButton_sine = QtWidgets.QPushButton(self.frame)
        self.pushButton_sine.setGeometry(QtCore.QRect(540, 70, 71, 47))
        self.pushButton_sine.setStyleSheet("background-color: rgb(150, 150, 150);\n"
"border :3px solid black")
        self.pushButton_sine.setText("")
        icon1 = QtGui.QIcon()
        icon1.addPixmap(QtGui.QPixmap("sine_wave_u223F_icon_256x256.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.pushButton_sine.setIcon(icon1)
        self.pushButton_sine.setIconSize(QtCore.QSize(40, 40))
        self.pushButton_sine.setCheckable(True)
        self.pushButton_sine.setObjectName("pushButton_sine")
        self.dial = QtWidgets.QDial(self.frame)
        self.dial.setGeometry(QtCore.QRect(40, 210, 111, 101))
        self.dial.setNotchesVisible(True)
        self.dial.setObjectName("dial")
        self.frame_6 = QtWidgets.QFrame(self.frame)
        self.frame_6.setGeometry(QtCore.QRect(30, 150, 441, 161))
        self.frame_6.setStyleSheet("background-color: rgb(150, 150, 150);")
        self.frame_6.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame_6.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame_6.setObjectName("frame_6")
        self.dial_2 = QtWidgets.QDial(self.frame_6)
        self.dial_2.setGeometry(QtCore.QRect(330, 70, 81, 81))
        self.dial_2.setStyleSheet("color:rgb(85, 85, 255)")
        self.dial_2.setNotchesVisible(True)
        self.dial_2.setObjectName("dial_2")
        self.dial_3 = QtWidgets.QDial(self.frame_6)
        self.dial_3.setGeometry(QtCore.QRect(180, 70, 101, 81))
        self.dial_3.setNotchesVisible(True)
        self.dial_3.setObjectName("dial_3")
        self.label_6 = QtWidgets.QLabel(self.frame_6)
        self.label_6.setGeometry(QtCore.QRect(20, 20, 101, 31))
        self.label_6.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")
        self.label_6.setObjectName("label_6")
        self.label_7 = QtWidgets.QLabel(self.frame_6)
        self.label_7.setGeometry(QtCore.QRect(180, 30, 101, 21))
        self.label_7.setStyleSheet("font: 13pt \"MS Shell Dlg 2\";")
        self.label_7.setObjectName("label_7")
        self.label_8 = QtWidgets.QLabel(self.frame_6)
        self.label_8.setGeometry(QtCore.QRect(330, 30, 91, 16))
        self.label_8.setStyleSheet("\n"
"font: 12pt \"MS Shell Dlg 2\";")
        self.label_8.setObjectName("label_8")
        self.label_9 = QtWidgets.QLabel(self.frame)
        self.label_9.setGeometry(QtCore.QRect(40, 30, 201, 21))
        self.label_9.setStyleSheet("font: 14pt \"MS Shell Dlg 2\";")
        self.label_9.setObjectName("label_9")
```

```
123         self.label_10 = QtWidgets.QLabel(self.frame)
124         self.label_10.setGeometry(QtCore.QRect(580, 20, 181, 31))
125         self.label_10.setStyleSheet("font: 14pt \"MS Shell Dlg 2\";
    ")
126         self.label_10.setObjectName("label_10")
127         self.pushButton_tringular = QtWidgets.QPushButton(self.
    frame)
128         self.pushButton_tringular.setGeometry(QtCore.QRect(700, 70,
     71, 47))
129         self.pushButton_tringular.setStyleSheet("background-color:
    rgb(150, 150, 150);\n"
130 "border :3px solid black")
131         self.pushButton_tringular.setText("")
132         icon2 = QtGui.QIcon()
133         icon2.addPixmap(QtGui.QPixmap("triangular-wave.png"), QtGui
    .QIcon.Normal, QtGui.QIcon.Off)
134         self.pushButton_tringular.setIcon(icon2)
135         self.pushButton_tringular.setIconSize(QtCore.QSize(40, 40))
136         self.pushButton_tringular.setCheckable(True)
137         self.pushButton_tringular.setChecked(False)
138         self.pushButton_tringular.setObjectName("
    pushButton_tringular")
139         self.frame_6.raise_()
140         self.pushButton_100k.raise_()
141         self.pushButton_100.raise_()
142         self.pushButton_1k.raise_()
143         self.pushButton_10k.raise_()
144         self.pushButton_10.raise_()
145         self.pushButton_1.raise_()
146         self.pushButton_square.raise_()
147         self.pushButton_sine.raise_()
148         self.dial.raise_()
149         self.label_9.raise_()
150         self.label_10.raise_()
151         self.pushButton_tringular.raise_()
152         self.frame_2 = QtWidgets.QFrame(Dialog)
153         self.frame_2.setGeometry(QtCore.QRect(60, 380, 1021, 421))
154         self.frame_2.setStyleSheet("background-color: rgb(84, 84,
    84);\n"
155 "border-radius: 25px;")
156         self.frame_2.setFrameShape(QtWidgets.QFrame.StyledPanel)
157         self.frame_2.setFrameShadow(QtWidgets.QFrame.Raised)
158         self.frame_2.setObjectName("frame_2")
159         self.frame_3 = QtWidgets.QFrame(self.frame_2)
160         self.frame_3.setGeometry(QtCore.QRect(810, 10, 201, 241))
161         self.frame_3.setStyleSheet("background-color: rgb(150, 150,
     150);\n"
162 "border-radius: 25px;")
163         self.frame_3.setFrameShape(QtWidgets.QFrame.StyledPanel)
164         self.frame_3.setFrameShadow(QtWidgets.QFrame.Raised)
165         self.frame_3.setObjectName("frame_3")
166         self.dial_5 = QtWidgets.QDial(self.frame_3)
167         self.dial_5.setGeometry(QtCore.QRect(50, 150, 101, 91))
168         self.dial_5.setNotchesVisible(True)
169         self.dial_5.setObjectName("dial_5")
170         self.dial_7 = QtWidgets.QDial(self.frame_3)
171         self.dial_7.setGeometry(QtCore.QRect(70, 40, 61, 71))
```

```
        self.dial_7.setNotchesVisible(True)
        self.dial_7.setObjectName("dial_7")
        self.label_2 = QtWidgets.QLabel(self.frame_3)
        self.label_2.setGeometry(QtCore.QRect(30, 10, 151, 31))
        self.label_2.setStyleSheet("font: 14pt \"MS Shell Dlg 2\";")
        self.label_2.setObjectName("label_2")
        self.label_4 = QtWidgets.QLabel(self.frame_3)
        self.label_4.setGeometry(QtCore.QRect(70, 120, 81, 31))
        self.label_4.setStyleSheet("font: 11pt \"MS Shell Dlg 2\";")
        self.label_4.setObjectName("label_4")
        self.frame_4 = QtWidgets.QFrame(self.frame_2)
        self.frame_4.setGeometry(QtCore.QRect(810, 270, 201, 131))
        self.frame_4.setStyleSheet("background-color: rgb(150, 150, 150);\n"
"border-radius: 25px;")
        self.frame_4.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame_4.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame_4.setObjectName("frame_4")
        self.trigger_level = QtWidgets.QDial(self.frame_4)
        self.trigger_level.setGeometry(QtCore.QRect(120, 50, 61, 71))
        self.trigger_level.setObjectName("trigger_level")
        self.label_11 = QtWidgets.QLabel(self.frame_4)
        self.label_11.setGeometry(QtCore.QRect(100, 20, 101, 31))
        self.label_11.setStyleSheet("font: 10pt \"MS Shell Dlg 2\";")
        self.label_11.setObjectName("label_11")
        self.radioButton = QtWidgets.QRadioButton(self.frame_4)
        self.radioButton.setGeometry(QtCore.QRect(10, 60, 111, 20))
        self.radioButton.setChecked(True)
        self.radioButton.setObjectName("radioButton")
        self.frame_5 = QtWidgets.QFrame(self.frame_2)
        self.frame_5.setGeometry(QtCore.QRect(630, 10, 161, 391))
        self.frame_5.setStyleSheet("background-color: rgb(150, 150, 150);\n"
"border-radius: 25px;")
        self.frame_5.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame_5.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame_5.setObjectName("frame_5")
        self.dial_4 = QtWidgets.QDial(self.frame_5)
        self.dial_4.setGeometry(QtCore.QRect(30, 280, 91, 101))
        self.dial_4.setNotchesVisible(True)
        self.dial_4.setObjectName("dial_4")
        self.dial_6 = QtWidgets.QDial(self.frame_5)
        self.dial_6.setGeometry(QtCore.QRect(50, 160, 50, 64))
        self.dial_6.setNotchesVisible(True)
        self.dial_6.setObjectName("dial_6")
        self.label = QtWidgets.QLabel(self.frame_5)
        self.label.setGeometry(QtCore.QRect(20, 130, 121, 21))
        self.label.setStyleSheet("font: 14pt \"MS Shell Dlg 2\";")
        self.label.setObjectName("label")
        self.label_5 = QtWidgets.QLabel(self.frame_5)
        self.label_5.setGeometry(QtCore.QRect(10, 20, 141, 31))
        self.label_5.setStyleSheet("font: 16pt \"MS Shell Dlg 2\";")
```

```
        self.label_5.setObjectName("label_5")
        self.label_3 = QtWidgets.QLabel(self.frame_5)
        self.label_3.setGeometry(QtCore.QRect(40, 240, 91, 31))
        self.label_3.setStyleSheet("font: 10pt \"MS Shell Dlg 2\";\n"
"font: 12pt \"MS Shell Dlg 2\";")
        self.label_3.setObjectName("label_3")
        self.widget = QtWidgets.QWidget(self.frame_2)
        self.widget.setGeometry(QtCore.QRect(20, 10, 581, 401))
        self.widget.setObjectName("widget")

        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "ECSP"))
        self.pushButton_100k.setText(_translate("Dialog", "100K"))
        self.pushButton_100.setText(_translate("Dialog", "100"))
        self.pushButton_1k.setText(_translate("Dialog", "1K"))
        self.pushButton_10k.setText(_translate("Dialog", "10K"))
        self.pushButton_10.setText(_translate("Dialog", "10"))
        self.pushButton_1.setText(_translate("Dialog", "1"))
        self.label_6.setText(_translate("Dialog", "Frequency"))
        self.label_7.setText(_translate("Dialog", "Amplitude"))
        self.label_8.setText(_translate("Dialog", "DC Offset"))
        self.label_9.setText(_translate("Dialog", "Frequency Select"))
        self.label_10.setText(_translate("Dialog", "Waveform Select"))
        self.label_2.setText(_translate("Dialog", "Horizontal Pos"))
        self.label_4.setText(_translate("Dialog", "Time/div"))
        self.label_11.setText(_translate("Dialog", "Trigger Level"))
        self.radioButton.setText(_translate("Dialog", "Rising / Falling"))
        self.label.setText(_translate("Dialog", "Vertical Pos"))
        self.label_5.setText(_translate("Dialog", "CHANNEL 1"))
        self.label_3.setText(_translate("Dialog", "Volts/div"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Dialog = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(Dialog)
    Dialog.show()
    sys.exit(app.exec_())
```

And also for controlling the generator and oscilloscope we have:

5.3 welcome page

```
import sys
```

```
import time

import matplotlib.pyplot as plt
import matplotlib as plot
from PyQt5 import QtWidgets

from matplotlib import animation
from pyqtgraph import PlotWidget, plot
import pyqtgraph as pg
import numpy as np
from scipy import signal
import gui2
import Main3
from PyQt5.QtWidgets import QDialog, QApplication, QWidget,
    QVBoxLayout
from matplotlib.figure import Figure
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg
from PyQt5 import QtCore, QtGui, QtWidgets


class WelcomePage(QWidget, gui2.Ui_Dialog):
    def __init__(self):
        super(WelcomePage, self).__init__()
        self.setupUi(self)
        self.start_button.clicked.connect(self.change_window)

    def change_window(self):
        stackwidget.setCurrentIndex(stackwidget.currentIndex()+1)


if __name__ == '__main__':
    app = QApplication(sys.argv)
    stackwidget = QtWidgets.QStackedWidget()
    wp = WelcomePage()
    w = Main3.MainWidget()
    stackwidget.setWindowTitle("ECSP")
    stackwidget.addWidget(wp)
    stackwidget.addWidget(w)
    stackwidget.setFixedHeight(834)
    stackwidget.setFixedWidth(1118)
    stackwidget.setStyleSheet("background-color: rgb(54, 54, 54);")
    stackwidget.show()
    sys.exit(app.exec_())
```

By this program we have make two different widgets for welcome page and also devices page and start by showing welcome page if you choose to see other page the other widget will be call.

5.4 control devices

```
import sys
import time

import matplotlib.pyplot as plt
import matplotlib as plot
from PyQt5 import QtWidgets

from matplotlib import animation
```

```
from pyqtgraph import PlotWidget, plot
import pyqtgraph as pg
import numpy as np
from scipy import signal

import Main
from PyQt5.QtWidgets import QDialog, QApplication, QWidget,
    QVBoxLayout
from matplotlib.figure import Figure
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg
from PyQt5 import QtCore, QtGui, QtWidgets

import gui


class MatplotlibWidget(QWidget):
    def __init__(self, parent=None):
        super(MatplotlibWidget, self).__init__(parent)
        self.figure = plt.figure()

        self.canvas = FigureCanvasQTAgg(self.figure)
        self.axis = self.figure.add_subplot(111)

        self.layoutvertical = QVBoxLayout(self)
        self.layoutvertical.addWidget(self.canvas)


class MainWidget(QWidget, gui.Ui_Dialog):
    def __init__(self):
        super(MainWidget, self).__init__()
        self.setupUi(self)
        self.wavename = 'sine'
        self.frequencyValue = 1
        self.rising = True
        self.matplotwidget = MatplotlibWidget()
        self.layoutvertical = QVBoxLayout(self.widget)
        self.layoutvertical.addWidget(self.matplotwidget)

        self.dial.setValue(50)
        self.dial_2.setValue(50)
        self.dial_3.setValue(50)
        self.dial_4.setValue(20)
        self.dial_5.setValue(50)
        self.dial_6.setValue(50)
        self.dial_7.setValue(50)

        # self.plot_widget()
        self.anim1 = animation.FuncAnimation(self.matplotwidget.
    figure, self.update_sine, frames=2000, interval=10,
                                             blit=False)
        self.anim2 = animation.FuncAnimation(self.matplotwidget.
    figure, self.update_square, frames=2000, interval=10,
                                             blit=False)
        self.anim3 = animation.FuncAnimation(self.matplotwidget.
    figure, self.update_triangular, frames=2000,
                                             interval=10,
                                             blit=False)
```

```
62
63          self.pushButton_sine.clicked.connect(self.
    changewave_to_sine)
64          self.pushButton_square.clicked.connect(self.
    changewave_to_square)
65          self.pushButton_tringular.clicked.connect(self.
    changewave_to_tri)
66
67          self.pushButton_1.clicked.connect(self.changefrequency_to_1
    )
68          self.pushButton_10.clicked.connect(self.
    changefrequency_to_10)
69          self.pushButton_100.clicked.connect(self.
    changefrequency_to_100)
70          self.pushButton_1k.clicked.connect(self.
    changefrequency_to_1k)
71          self.pushButton_10k.clicked.connect(self.
    changefrequency_to_10k)
72          self.pushButton_100k.clicked.connect(self.
    changefrequency_to_100k)
73
74          self.radioButton.toggled.connect(self.set_rise)
75
76          # self.dial.valueChanged.connect(self.plot_widget)
77          # self.dial_3.valueChanged.connect(self.plot_widget)
78          # self.dial_2.valueChanged.connect(self.plot_widget)
79          self.dial_4.valueChanged.connect(self.plot)
80          self.dial_5.valueChanged.connect(self.plot)
81          self.trigger_level.valueChanged.connect(self.plot)
82          # self.dial_6.valueChanged.connect(self.plot_widget)
83          # self.dial_7.valueChanged.connect(self.plot_widget)
84
85          self.plot()
86
87      def plot(self):
88          z = 0.000005 * np.exp(15 * self.dial_5.value() / 100)
89          if self.wavename == 'sine':
90              self.anim2.pause()
91              self.anim3.pause()
92              self.matplotwidget.axis.clear()
93              self.matplotwidget.axis.set_facecolor("black")
94              self.matplotwidget.axis.grid()
95              self.x = np.linspace(-z, z, 1000)
96              self.line, = self.matplotwidget.axis.plot([], [], '
    yellow')
97              self.anim1.resume()
98
99          if self.wavename == 'square':
100             self.anim1.pause()
101             self.anim3.pause()
102             self.matplotwidget.axis.clear()
103             self.matplotwidget.axis.grid()
104             self.matplotwidget.axis.set_facecolor("black")
105             self.x = np.linspace(-z, z, 1000)
106             self.line, = self.matplotwidget.axis.plot([], [], '
    yellow')
107             self.anim2.resume()
```

```
108
109         if self.wavename == 'tri':
110             self.anim1.pause()
111             self.anim2.pause()
112             self.matplotwidget.axis.clear()
113             self.matplotwidget.axis.set_facecolor("black")
114             self.matplotwidget.axis.grid()
115             self.x = np.linspace(-z, z, 1000)
116             self.line, = self.matplotwidget.axis.plot([], [], '
    yellow')
117             self.anim3.resume()
118
119     def update_sine(self, i):
120         z = 0.000005 * np.exp(15 * self.dial_5.value() / 100)
121         if self.chech_trigger():
122             self.matplotwidget.axis.set_ylim([-self.dial_4.value(),
     self.dial_4.value()])
123             self.matplotwidget.axis.set_xlim([-z, z])
124             self.matplotwidget.axis.axhline(y=50 - self.
    trigger_level.value())
125             self.line.set_xdata(np.linspace(-z, z, 1000))
126             self.line.set_ydata(
127                 self.dial_3.value() / 10 * np.sin(2 * np.pi * self.
    frequencyValue * (self.dial.value() / 100) * (
128                         self.x - self.dial_7.value() / 10) + i /
    10.0) - 50 + self.dial_2.value() - 50 + self.dial_6.value())  #
     update the data
129
130         else:
131
132             if self.rising:
133                 self.matplotwidget.axis.set_ylim([-self.dial_4.
    value(), self.dial_4.value()])
134                 self.matplotwidget.axis.set_xlim([-z, z])
135                 self.matplotwidget.axis.axhline(y=50 - self.
    trigger_level.value())
136                 self.line.set_xdata(np.linspace(-z, z, 1000))
137                 self.line.set_ydata(
138                     self.dial_3.value() / 10 * np.sin(2 * np.pi *
    self.frequencyValue * (self.dial.value() / 100) * (
139                             self.x - self.
    get_trigger_value_sine_rising() - self.dial_7.value() / 10)) -
    50 + self.dial_2.value() - 50 + self.dial_6.value())  # update
    the data
140
141
142
143             else:
144                 self.matplotwidget.axis.set_ylim([-self.dial_4.
    value(), self.dial_4.value()])
145                 self.matplotwidget.axis.set_xlim([-z, z])
146                 self.matplotwidget.axis.axhline(y=50 - self.
    trigger_level.value())
147                 self.line.set_xdata(np.linspace(-z, z, 1000))
148                 self.line.set_ydata(
149                     self.dial_3.value() / 10 * np.sin(2 * np.pi *
    self.frequencyValue * (self.dial.value() / 100) * (
```

```
                            self.x + (1 / (2 * self.frequencyValue
* (
                            self.dial.value() / 100))) - self.
get_trigger_value_sine_falling() - self.dial_7.value() / 10)) -
 50 + self.dial_2.value() - 50 + self.dial_6.value())  # update
 the data

        return self.line,

    def update_square(self, i):
        z = 0.000005 * np.exp(15 * self.dial_5.value() / 100)
        if self.chech_trigger():
            self.matplotwidget.axis.set_ylim([-self.dial_4.value(),
 self.dial_4.value()])
            self.matplotwidget.axis.set_xlim([-z, z])
            self.matplotwidget.axis.axhline(y=50 - self.
trigger_level.value())
            self.line.set_xdata(np.linspace(-z, z, 1000))
            self.line.set_ydata(
                self.dial_3.value() / 10 * signal.square(2 * np.pi
* self.frequencyValue * (self.dial.value() / 100) * (
                        self.x - self.dial_7.value() / 10) + i /
10.0,
                                                             0.5) - 50
+ self.dial_2.value() - 50 + self.dial_6.value())  # update the
 data

        else:

            if self.rising == True:
                self.matplotwidget.axis.set_ylim([-self.dial_4.
value(), self.dial_4.value()])
                self.matplotwidget.axis.set_xlim([-z, z])
                self.matplotwidget.axis.axhline(y=50 - self.
trigger_level.value())
                self.line.set_xdata(np.linspace(-z, z, 1000))
                self.line.set_ydata(
                    self.dial_3.value() / 10 * signal.square(
                        2 * np.pi * self.frequencyValue * (self.
dial.value() / 100) * (
                                self.x - 1 / (
                                2 * self.frequencyValue * (self.
dial.value() / 100)) - self.dial_7.value() / 10),
                        0.5) - 50 + self.dial_2.value() - 50 + self
.dial_6.value())  # update the data


            else:
                self.matplotwidget.axis.set_ylim([-self.dial_4.
value(), self.dial_4.value()])
                self.matplotwidget.axis.set_xlim([-z, z])
                self.matplotwidget.axis.axhline(y=50 - self.
trigger_level.value())
                self.line.set_xdata(np.linspace(-z, z, 1000))
                self.line.set_ydata(
                    self.dial_3.value() / 10 * signal.square(
```

```
                        2 * np.pi * self.frequencyValue * (self.
dial.value() / 100) * (
                                self.x - self.dial_7.value() / 10))
 - 50 + self.dial_2.value() - 50 + self.dial_6.value())  #
update the data

        return self.line,

    def update_triangular(self, i):
        z = 0.000005 * np.exp(15 * self.dial_5.value() / 100)
        if self.chech_trigger():
            self.matplotwidget.axis.set_ylim([-self.dial_4.value(),
 self.dial_4.value()])
            self.matplotwidget.axis.set_xlim([-z, z])
            self.matplotwidget.axis.axhline(y=50 - self.
trigger_level.value())
            self.line.set_xdata(np.linspace(-z, z, 1000))
            self.line.set_ydata(
                self.dial_3.value() / 10 * signal.sawtooth(
                    2 * np.pi * self.frequencyValue * (self.dial.
value() / 100) * (
                            self.x - self.dial_7.value() / 10) + i
/ 10.0,
                    0.5) - 50 + self.dial_2.value() - 50 + self.
dial_6.value())  # update the data

        else:

            if self.rising:
                self.matplotwidget.axis.set_ylim([-self.dial_4.
value(), self.dial_4.value()])
                self.matplotwidget.axis.set_xlim([-z, z])
                self.matplotwidget.axis.axhline(y=50 - self.
trigger_level.value())
                self.line.set_xdata(np.linspace(-z, z, 1000))
                self.line.set_ydata(
                    self.dial_3.value() / 10 * signal.sawtooth(
                        2 * np.pi * self.frequencyValue * (self.
dial.value() / 100) * (
                                self.x - self.
get_trigger_value_tri_rising() - self.dial_7.value() / 10),
                        0.5) - 50 + self.dial_2.value() - 50 + self
.dial_6.value())  # update the data









            else:
                self.matplotwidget.axis.set_ylim([-self.dial_4.
value(), self.dial_4.value()])
                self.matplotwidget.axis.set_xlim([-z, z])
                self.matplotwidget.axis.axhline(y=50 - self.
trigger_level.value())
                self.line.set_xdata(np.linspace(-z, z, 1000))
                self.line.set_ydata(
                    self.dial_3.value() / 10 * signal.sawtooth(
```

```
                        2 * np.pi * self.frequencyValue * (self.
    dial.value() / 100) * (
                                self.x + (1 / (2 * self.
    frequencyValue * (
                                self.dial.value() / 100))) - self.
    get_trigger_value_tri_falling() - self.dial_7.value() / 10),
                        0.5) - 50 + self.dial_2.value() - 50 + self
    .dial_6.value())  # update the data


        return self.line,

    def Color_of_w(self):
        self.pushButton_sine.setStyleSheet("background-color : Gray
    ;\n"
"border :3px solid black")
        self.pushButton_tringular.setStyleSheet("background-color :
     Gray;\n"
"border :3px solid black")
        self.pushButton_square.setStyleSheet("background-color :
    Gray;\n"
"border :3px solid black")

    def changewave_to_sine(self):
        self.Color_of_w()
        self.pushButton_sine.setStyleSheet("background-color : red
    ;\n"
"border :3px solid black")
        self.wavename = 'sine'
        self.plot()

    def changewave_to_square(self):
        self.Color_of_w()
        self.pushButton_square.setStyleSheet("background-color :
    red;\n"
"border :3px solid black")
        self.wavename = 'square'
        self.plot()

    def changewave_to_tri(self):
        self.Color_of_w()
        self.pushButton_tringular.setStyleSheet("background-color :
     red;\n"
"border :3px solid black")
        self.wavename = 'tri'
        self.plot()

    def Color_of_f(self):
        self.pushButton_1.setStyleSheet("background-color : Gray;\n
    "
"border :3px solid black")
        self.pushButton_10.setStyleSheet("background-color : Gray;\
    n"
"border :3px solid black")
        self.pushButton_100.setStyleSheet("background-color : Gray
    ;\n"
"border :3px solid black")
```

```
276         self.pushButton_1k.setStyleSheet("background-color : Gray;\
    n"
277 "border :3px solid black")
278         self.pushButton_10k.setStyleSheet("background-color : Gray
    ;\n"
279 "border :3px solid black")
280         self.pushButton_100k.setStyleSheet("background-color : Gray
    ;\n"
281 "border :3px solid black")
282 
283     def changefrequency_to_1(self):
284         self.Color_of_f()
285         self.pushButton_1.setStyleSheet("background-color : red;\n"
286 "border :3px solid black")
287         self.frequencyValue = 1
288         self.plot()
289 
290     def changefrequency_to_10(self):
291         self.Color_of_f()
292         self.pushButton_10.setStyleSheet("background-color : red;\n
    "
293 "border :3px solid black")
294         self.frequencyValue = 10
295         self.plot()
296 
297     def changefrequency_to_100(self):
298         self.Color_of_f()
299         self.pushButton_100.setStyleSheet("background-color : red;\
    n"
300 "border :3px solid black")
301         self.frequencyValue = 100
302         self.plot()
303 
304     def changefrequency_to_1k(self):
305         self.Color_of_f()
306         self.pushButton_1k.setStyleSheet("background-color : red;\n
    "
307 "border :3px solid black")
308         self.frequencyValue = 1000
309         self.plot()
310 
311     def changefrequency_to_10k(self):
312         self.Color_of_f()
313         self.pushButton_10k.setStyleSheet("background-color : red;\
    n"
314 "border :3px solid black")
315         self.frequencyValue = 10000
316         self.plot()
317 
318     def changefrequency_to_100k(self):
319         self.Color_of_f()
320         self.pushButton_100k.setStyleSheet("background-color : red
    ;\n"
321 "border :3px solid black")
322         self.frequencyValue = 100000
323         self.plot()
324 
```

```
325     def chech_trigger(self):
326         if 50 - self.trigger_level.value() > self.dial_3.value() /
    10 - 50 + self.dial_2.value() - 50 + self.dial_6.value() or 50
    - self.trigger_level.value() < -self.dial_3.value() / 10 - 50 +
     self.dial_2.value() - 50 + self.dial_6.value():
327             return True
328 
329         else:
330             return False
331 
332     def get_trigger_value_sine_rising(self):
333         ans = (100 / (2 * np.pi * self.frequencyValue * self.dial.
    value())) * np.arcsin((10 / self.dial_3.value()) * (
334                 50 - self.trigger_level.value() + 100 - self.dial_2
    .value() - self.dial_6.value())) + self.dial_7.value() / 10
335         return -ans
336 
337     def get_trigger_value_sine_falling(self):
338         ans = (100 / (2 * np.pi * self.frequencyValue * self.dial.
    value())) * np.arcsin((10 / self.dial_3.value()) * (
339                  - 50 + self.trigger_level.value() + 100 - self.
    dial_2.value() - self.dial_6.value())) + self.dial_7.value() /
    10
340         return -ans
341 
342     def get_trigger_value_tri_falling(self):
343         ans = (100 / (2 * np.pi * self.frequencyValue * self.dial.
    value())) * self.arc_triangular(
344             (10 / self.dial_3.value()) * (
345                     - 50 + self.trigger_level.value() + 100 - self.
    dial_2.value() - self.dial_6.value())) + self.dial_7.value() /
    10
346         return -ans
347 
348     def get_trigger_value_tri_rising(self):
349         ans = (100 / (2 * np.pi * self.frequencyValue * self.dial.
    value())) * self.arc_triangular(
350             (10 / self.dial_3.value()) * (
351                     50 - self.trigger_level.value() + 100 - self.
    dial_2.value() - self.dial_6.value())) + self.dial_7.value() /
    10
352         return -ans
353 
354     def arc_triangular(self, x):
355         return (np.pi / 2) * (x + 1)
356 
357     def set_rise(self):
358         if self.radioButton.isChecked():
359             self.rising = True
360             self.plot()
361 
362         else:
363             self.rising = False
364             self.plot()
365 
366 
367 if __name__ == '__main__':
```

```
368     app = QApplication(sys.argv)
369     w = MainWidget()
370     w.show()
371     sys.exit(app.exec_())
```

if I want to describe the code by it's important functions we will have the program like this:

5.4.1 class MatplotlibWidget line 23

It is for preparing page to plot signals.

5.4.2 init function of MainWidget class line 35

This function initialize the class to generate signals by the situation is chose by user.

5.4.3 plot function of MainWidget class line 87

This function plot the wave in chose interval by using animation tool of python.

5.4.4 update-sine function of MainWidget class line 119

This function called in every loop and check the position of sinusoidal signal and change the values as user wants to plot the wanted wave.

5.4.5 update-square function of MainWidget class line 155

This function called in every loop and check the position of square signal and change the values as user wants to plot the wanted wave.

5.4.6 update-triangular function of MainWidget class line 195

This function called in every loop and check the position of sinusoidal signal and change the values as user wants to plot the wanted wave.

5.4.7 Color-of-w and changewave-to-specialWave functions of MainWidget class in lines 240-248-255-262

These functions are control and change the color of buttons.

5.4.8 changefrequency-to-n functions of MainWidget class in lines 283-290-297-304-311-318

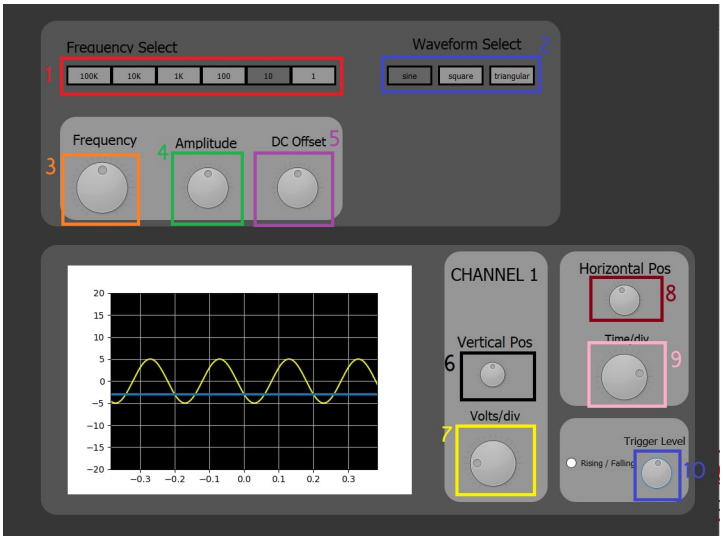
These functions are control and change amount of signals frequency.

5.4.9 trigger functions of MainWidget class

These functions are control and change trigger level and also position(rising or falling) and also give proper amounts to plot function to plot the wave.

5.5 show functions relation with gui

if I want to describe the functions of the program we will have the GUI like this:



5.5.1 1.Frequency select

From line 283 to 323 we control buttons on the other world we only choose order of frequency.Then plot the new signal.

5.5.2 2.Waveform select

From line 240 to 267 we use the buttons to change wave form.
And also how does every signal plotted?
we use lines 87 - 238 to do this(by plot and update functions) for example how does the sinusoidal signal was generated:
it happens by change the wave name to sine in line 253 then the plot function start to change in line 87 start to change the wave form and plot it.

5.5.3 3,4,5,6,7,8,9.Frequency,amplitude,DC offset,vertical pos , time/-div

from line 119 to 238 inside functions we can change frequency amount or other ones then by changing amounts the signal was shown will be change.
10.Trigger level:
from line 325 to 364 we have some special functions for different signals to check that the situation we choose for trigger is possible or not then it will show the signal.

6 performance

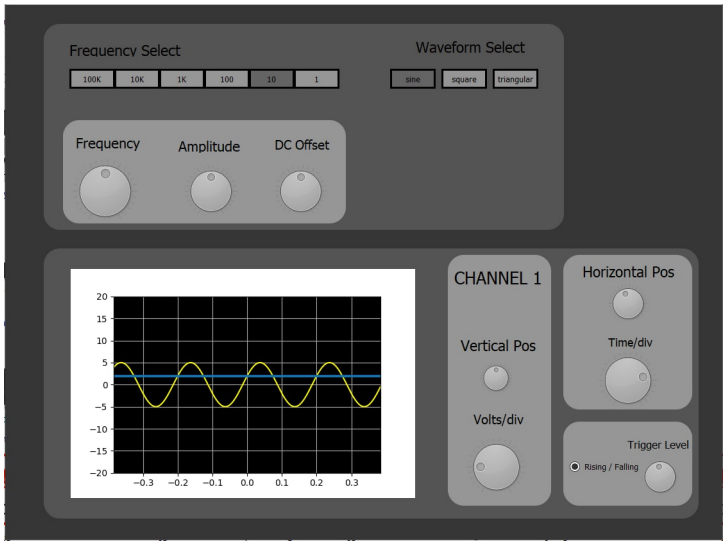


Figure 3: sinusoidal signal with frequency 10 and rising trigger level.

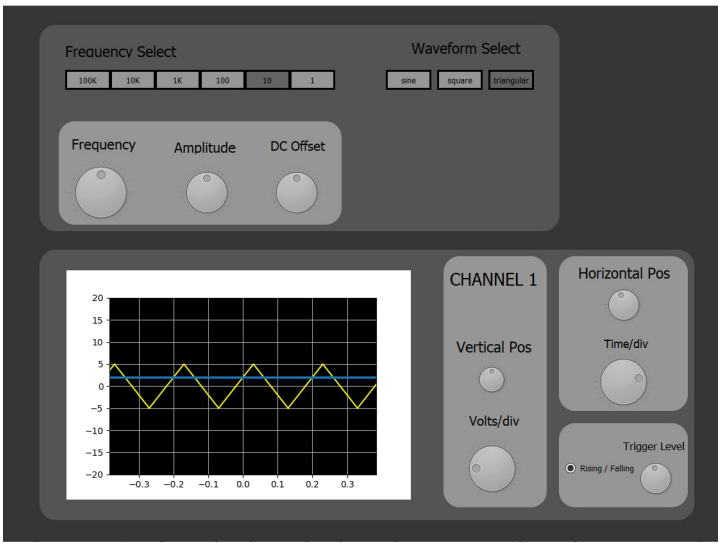


Figure 4: triangular signal with frequency 10 and rising trigger level.

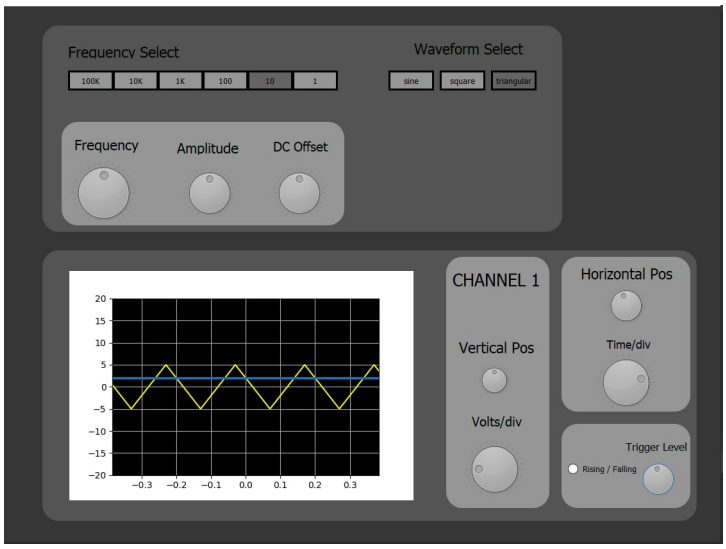


Figure 5: triangular signal with frequency 10 and falling trigger level.

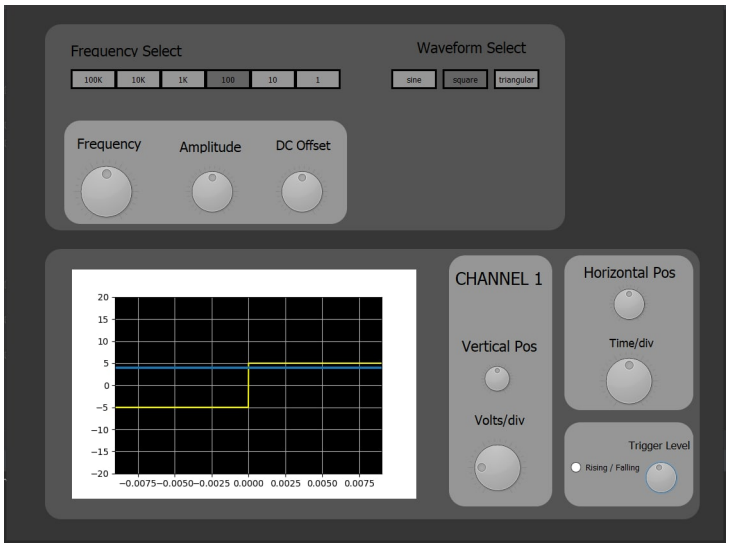


Figure 6: square signal with frequency 100.

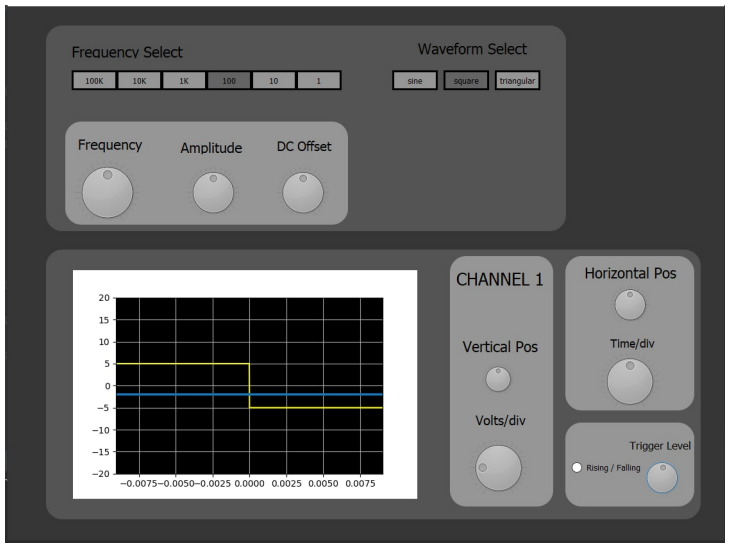


Figure 7: square signal with frequency 100.