آزمایشگاه سخت‌افزار

فاز سوم پروژه لاجیک آنالیزر
نرم‌افزاری و تستر دیجیتال

استاد: دکتر اجلالی

تیم شماره ۷ ـ اعضای تیم:

مهرداد صابری ـ ۹۷۱۱۰۱۳۳

محمد مهدوی ـ ۹۷۱۱۰۲۲۸

مسیح اسکندر ـ ۹۷۱۰۵۷۳۶

۱ هدف فاز و کارهای انجام شده

هدف این فاز اضافه کردن قابلیت ذخیره‌سازی سیگنال‌های موجود در نرم‌افزار در فایل دلخواه، و بازیابی یک سیگنال از فایل ذخیره‌شده می‌باشد. این قابلیت به نحوی پیاده‌سازی شده‌است که کاربر بتواند هریک از ۱۶ سیگنال موجود در نرم‌افزار را انتخاب و در فایل ذخیره کند. همچنین با بازیابی یک سیگنال ذخیره‌شده، آن سیگنال به لیست سیگنال‌های نمایش‌داده شده در نرم‌افزار اضافه می‌شود. علاوه بر این قابلیت حذف آخرین سیگنال در حال نمایش از لیست سیگنال‌ها نیز وجود دارد تا بتوانیم سیگنال‌های بازیابی شده را حذف کنیم.

۲ کدها

این فاز نیز مانند فاز سوم با استفاده از پایتون و کتابخانه‌ی گرافیکی tkinter پیاده‌سازی شده است. کدهای زده شده به فایل کد فاز دوم اضافه شده‌اند و قابلیت ذخیره‌سازی و بازیابی به همراه محیط کاربری‌اش به نرم‌افزار قبلی افزوده گردیده. حال به بررسی کلی بخش‌های اضافه شده در کد در فاز سوم می‌پردازیم.

```
def set_save_buttons():
    def save_signal(ind):
        nums = signal_array[ind][:]
        files = [('All Files', '*.*'), ('Text Document', '*.txt')]
        file = asksaveasfile(filetypes=files, defaultextension=files)
        if file:
            for num in nums:
                file.write(str(num) + '\n')
            file.close()
    for i in range(N):
        if i < N // 2:
            y = height - 200
        else:
            y = height - 100
        x = (i % (N // 2)) * (width - 400) / (N // 2 - 1) + 300
        btn = Button(window, text='💾', command=lambda: save_signal(
            i), bg='#0052cc', fg='#ffffff', font=font.Font(size=12))
        btn.place(x=x, y=y)
```

تکه کد بالا جهت ساختن و تعریف کاربری دکمه‌های ذخیره‌سازی در برنامه است. برای اینکار به تعداد سیگنال‌های ورودی برنامه که ۱۶ تا است دکمه ساخته می‌شود و با کلیک روی هرکدام تابع save_signal برای سیگنال مربوطه صدا زده می‌شود. در این تابع از کاربر درخواست محل ذخیره و نام فایل می‌شود و مقادیر سیگنال مربوطه که به صورت آرایه در برنامه ذخیره کرده‌ایم در فایل مربوطه ذخیره‌سازی می‌شوند. دقت می‌کنیم که سیگنال‌های ورودی به صورت زنده‌اند و هر لحظه به انتهای آرایه‌ی هر سیگنال اضافه می‌شود. اما وقتی سیگنالی را ذخیره می‌کنیم مقادیر آن تا این لحظه ذخیره می‌شوند و بعدا می‌توانیم این مقادیر را بازیابی کنیم.

```
load_button = Button(window, text="Load", command=load_signal,
                     bg='#0052cc', fg='#ffffff', font=font.Font(size=20))

unload_button = Button(window, text="Unload", command=unload_signal,
                     bg='#0052cc', fg='#ffffff', font=font.Font(size=20))
load_button.place(x=50, y=height // 2 - 500, height=100, width=100)
unload_button.place(x=50, y=height // 2 - 300, height=100, width=100)
```

تکه کد بالا دو دکمه‌ی load برای بازیابی یک فایل و نمایش سیگنالش، و unload برای پاک کردن آخرین سیگنال بازیابی شده از لیست سیگنال‌های در حال نمایش را می‌سازد. هرکدام از این دکمه‌ها در صورت کلیک شدن تابع مخصوصشان را صدا می‌زنند.

```
    def load_signal():
        global thread_lock
        thread_lock.acquire()
        file = askopenfile()
        if file:
            sig = []
            for line in file.readlines():
                num = int(line.strip())
                sig.append(num)
            if len(sig) < MAX_L:
                sig = [0] * (MAX_L - len(sig)) + sig
            loaded_signals.append(sig)
        thread_lock.release()
        init_plot()
```

تابع بالا هنگام کلیک روی دکمه‌ی load صدا زده می‌شود که از کاربر آدرس یک فایل را می‌گیرد و سیگنال ذخیره شده در این فایل را به آرایه‌ی loaded_signals اضافه می‌کند. سپس تابع init_plot

صدا زده می‌شود تا تمامی سیگنال‌های از نو نمایش داده شوند و سیگنال جدید نیز به دلیل حضور در آرایه‌ی loaded_signals نمایش داده می‌شود. دقت می‌کنیم که سیگنال‌های بازیابی شده دیگر به صورت زنده نیستند و مقادیر ثابتی دارند که در نمودار تغییر نمی‌کنند (برخلاف ۱۶ سیگنال ورودی که زنده‌اند).

```
    def unload_signal():
        global thread_lock, loaded_signals
        thread_lock.acquire()
        if len(loaded_signals):
            loaded_signals = loaded_signals[:-1]
        thread_lock.release()
        init_plot()
```

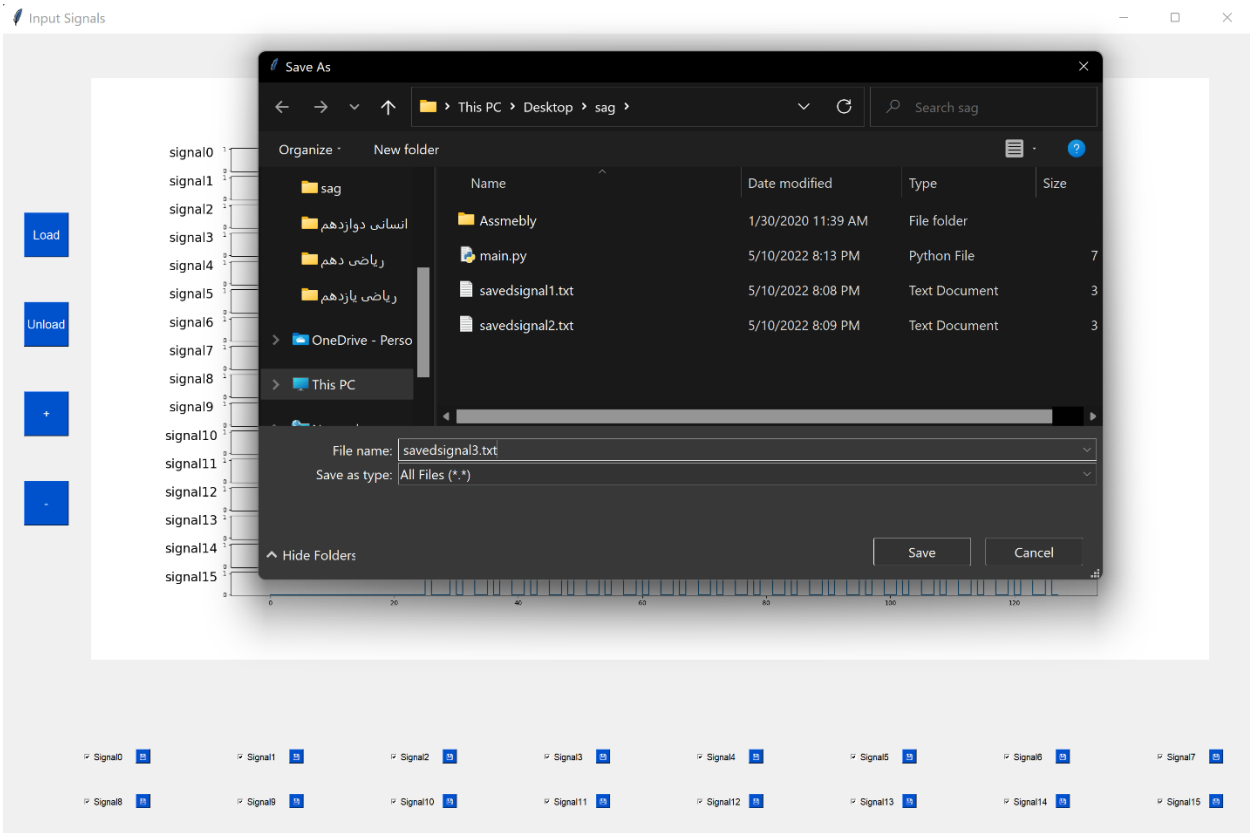
در صورت کلیک روی دکمه‌ی unload تابع بالا اجرا می‌شود که در صورت خالی نبودن آرایه‌ی loaded_signals، عنصر آخر آنرا حذف می‌کند و init_plot را صدا می‌کند تا نمودار سیگنال‌ها دوباره ساخته‌شود.

```
for i in range(len(loaded_signals)):
        axs_step += [axs[ind].step(x, loaded_signals[i][-L:])[0]]
        axs[ind].set_ylabel("Loaded signal{}".format(
            i), rotation=0, fontsize=20, labelpad=50,
horizontalalignment="center")
        ind += 1
```

این بخش از کد نیز در تابع init_plot است که سیگنال‌های موجود در آرایه‌ی loaded_signals را به نمودارهای نمایش داده‌شده اضافه می‌کند.

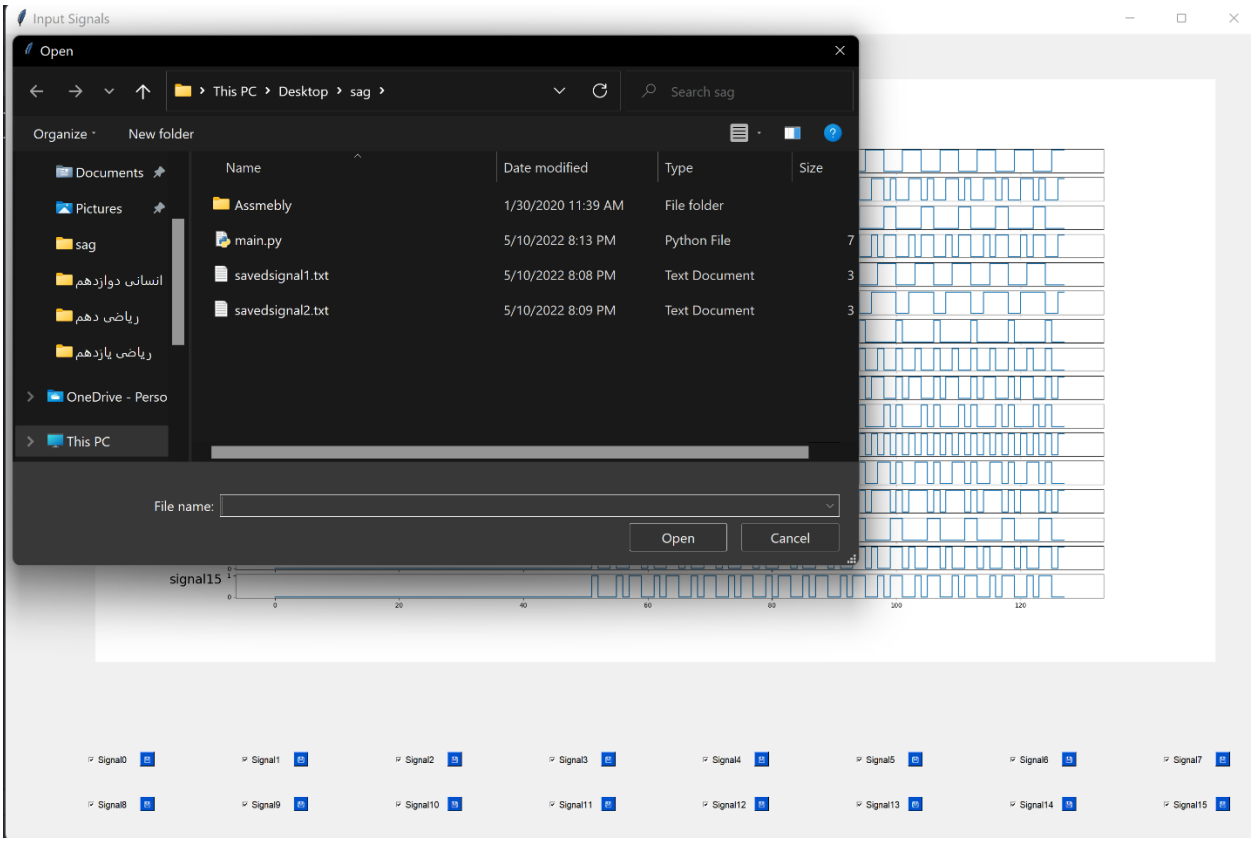
۳ محیط کاربری جدید نرم‌افزار

شکل 1 - دکمه‌های ذخیره‌سازی



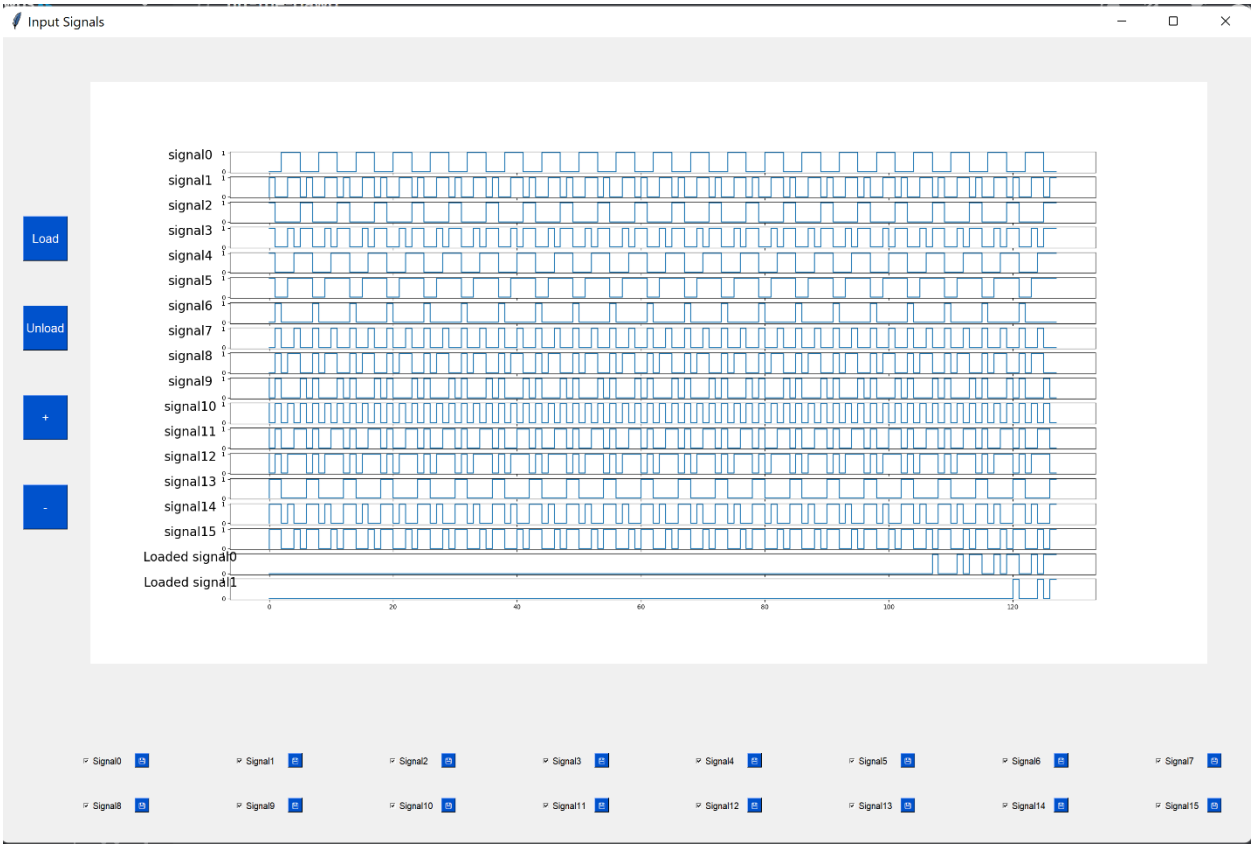
شکل 2 - انتخاب فایل برای ذخیره‌ی سیگنال

دکمه‌های ذخیره‌سازی همانطور که در شکل ۱ مشاهده می‌شود در کنار checkbox نمایش یا عدم نمایش هر سیگنال قرار گرفته‌اند. با کلیک روی این دکمه‌ها صفحه‌ای مانند شکل ۲ نمایان می‌شود که کاربر در آن فایل مورد نظرش برای ذخیره را انتخاب کند.



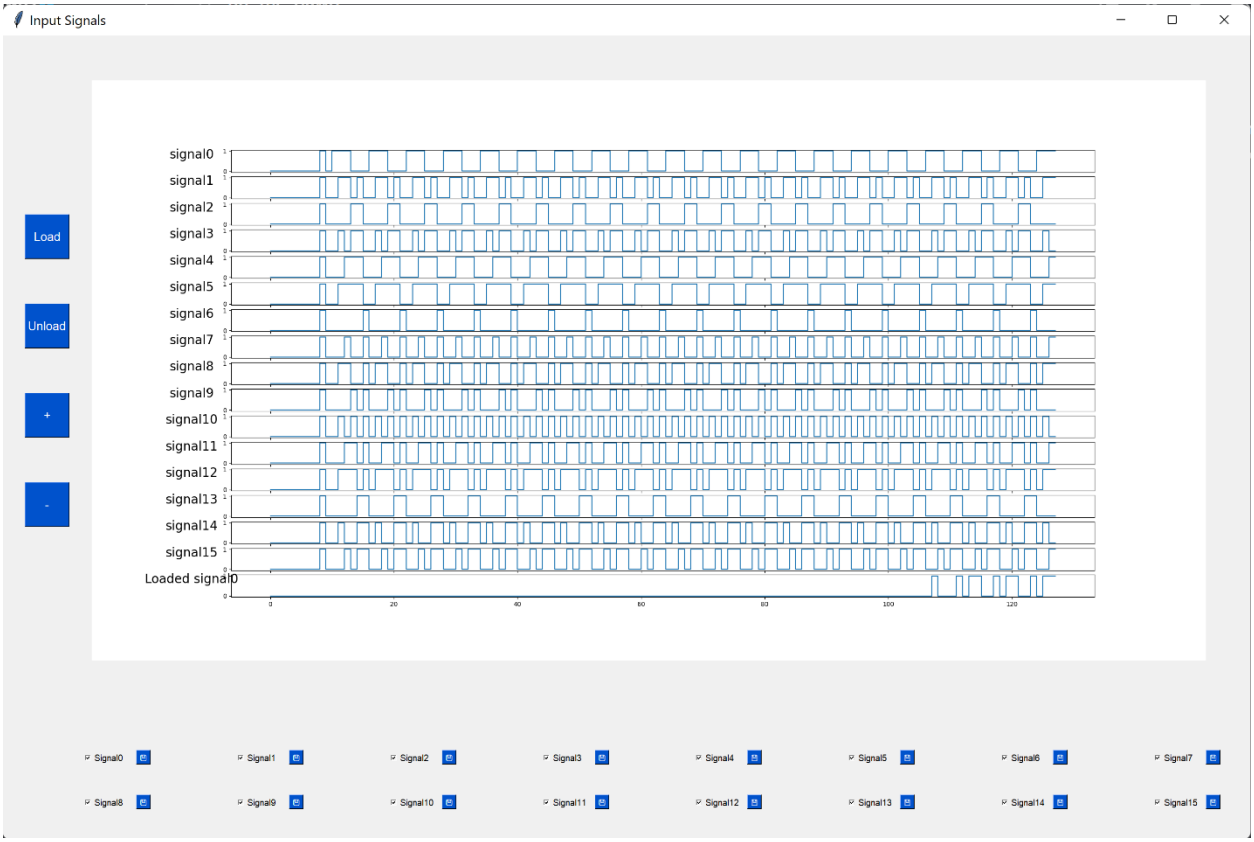
شکل 3 – انتخاب فایل برای بازیابی سیگنال

با کلیک کردن روی دکمه‌ی load نیز محیطی مشابه حالت save باز می‌شود که کاربر فایل مورد نظر برای بازیابی سیگنال را از آن انتخاب کند که این قابلیت را در شکل ۳ می‌بینید.



شکل 4 - نمایش سیگنال‌های بازیابی‌شده

سیگنال‌های بازیابی شده در انتهای لیست سیگنال‌ها نمایش داده می‌شوند که این مورد در شکل ۴ قابل مشاهده است.



شکل 5 - حذف آخرین سیگنال بازیابی‌شده

در شکل ۵ نیز loaded_signal1 که در شکل ۴ بازیابی‌شده و نمایش داده‌شده بود با دکمه‌ی unload حذف شده‌است.