آزمایشگاه سخت‌افزار

فاز دو 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 کلی ما در این فاز رسم نمودار سیگنال‌های آردوینو و پیاده‌سازی قابلیت زوم و اضافه و حذف کردن نمودار سیگنال‌ها بود. برای این در قسمت نرم‌افزاری رابط کاربری‌ای پیاده‌سازی کردیم که در آن سیگنال‌های دریافت شده نمایش داده شوند و امکان انتخاب سیگنال‌ها و زوم کردن روی آن‌ها وجود داشته باشد.

۲ برنامه‌ها

ابتدا در قسمت نرم‌افزاری با استفاده از Python و کتاب‌خانه Tkinter یک رابط کاربری برای نمایش سیگنال‌ها پیاده‌سازی کردیم و در این رابط کاربری با استفاده از کتابخانه matplotlib سیگنال‌ها را رسم کردیم. کد مربوطه در مخزن ما در دو نسخه موجود است که نسخه اول با ورودی های تصادفی و مستقل از آردوینو کار می‌کند و نسخه دوم برای کار با آردوینو است. تصویر زیر شکل کلی این رابط کاربری را نشان می‌دهد:

شکل ۱ شمای کلی رابط کاربری نرم‌افزاری

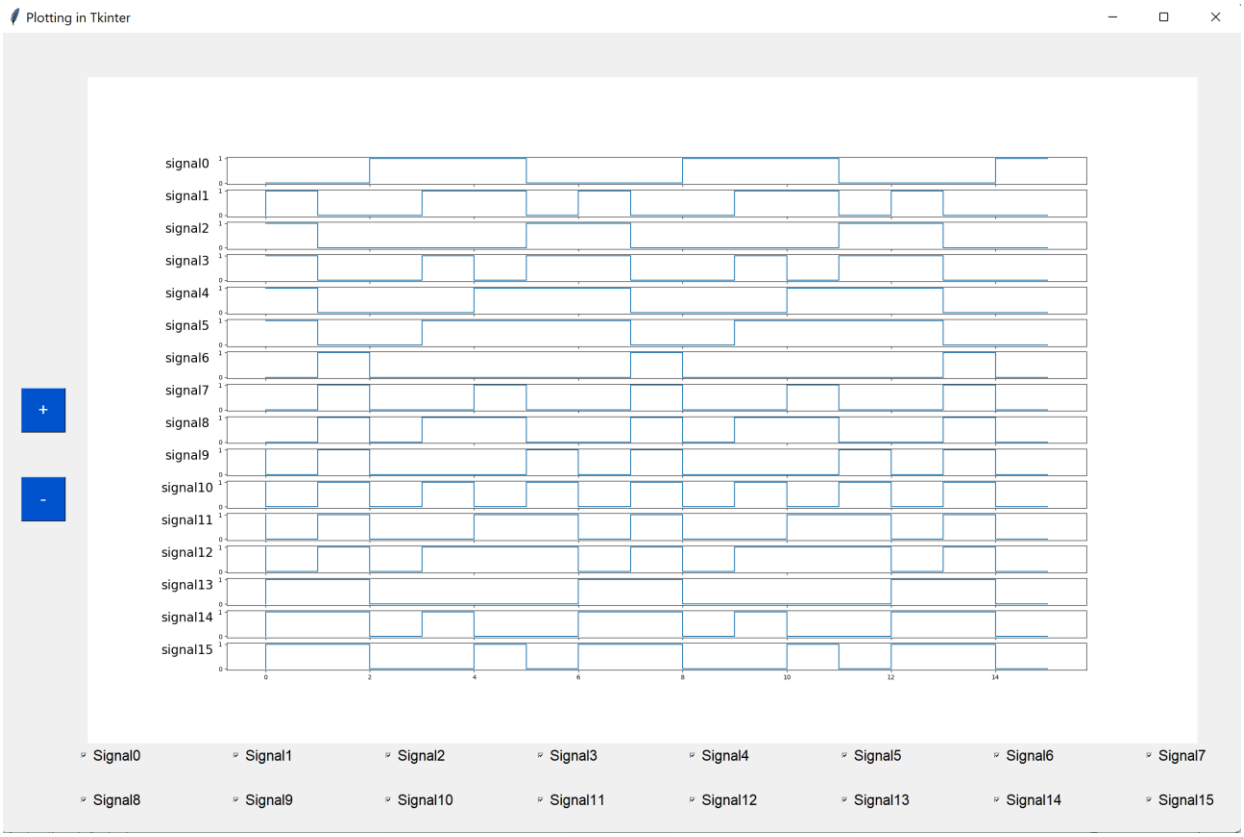
در این محیط گرافیکی نرم‌افزار می‌توانیم ۱۶ سیگنال ورودی را به صورت زنده مشاهده کنیم. در هر ثانیه، با گرفتن ورودی از آردوینو به هر سیگنال یک مقدار جدید اضافه می‌شود و مقدار آن به‌روز می‌شود. تکه کد زیر قسمت اصلی مربوط به نمایش نمودارها در رابط کاربری را نشان می‌دهد:

```
def init_plot():
    global fig, axs, canvas, axs_step, thread_lock
    thread_lock.acquire()
    if canvas:
        for item in canvas.get_tk_widget().find_all():
            canvas.get_tk_widget().delete(item)
    M = sum(shown_signals)
    fig, axs = plt.subplots(M, sharex=True, figsize=(25, 15))
    #for i in range(M):
    #    axs[i].get_yaxis().set_visible(False)
    x = [i for i in range(L)]
    ind = 0
    axs_step = []
    for i in range(N):
        if not shown_signals[i]:
            continue
        axs_step += [axs[ind].step(x, signal_array[i][-L:])[0]]
        axs[ind].set_ylabel("signal{}".format(i), rotation=0, fontsize=20, labelp
ad=50, horizontalalignment="center")
        ind += 1

    canvas = FigureCanvasTkAgg(fig,
                               master = window)


    canvas.draw()
    # placing the canvas on the Tkinter window
    canvas.get_tk_widget().place(x=200, y=100)
    thread_lock.release()
```

در این محیط در سمت چپ تصویر دکمه‌های زوم را مشاهده می‌کنید که با استفاده از آن‌ها می‌توانیم روی بازه‌های زمانی سیگنال‌ها Zoom in و Zoom out کنیم. برای مثال اگر چندین بار Zoom in کنیم نمودار سیگنال‌ها به شکل زیر در می‌آید:



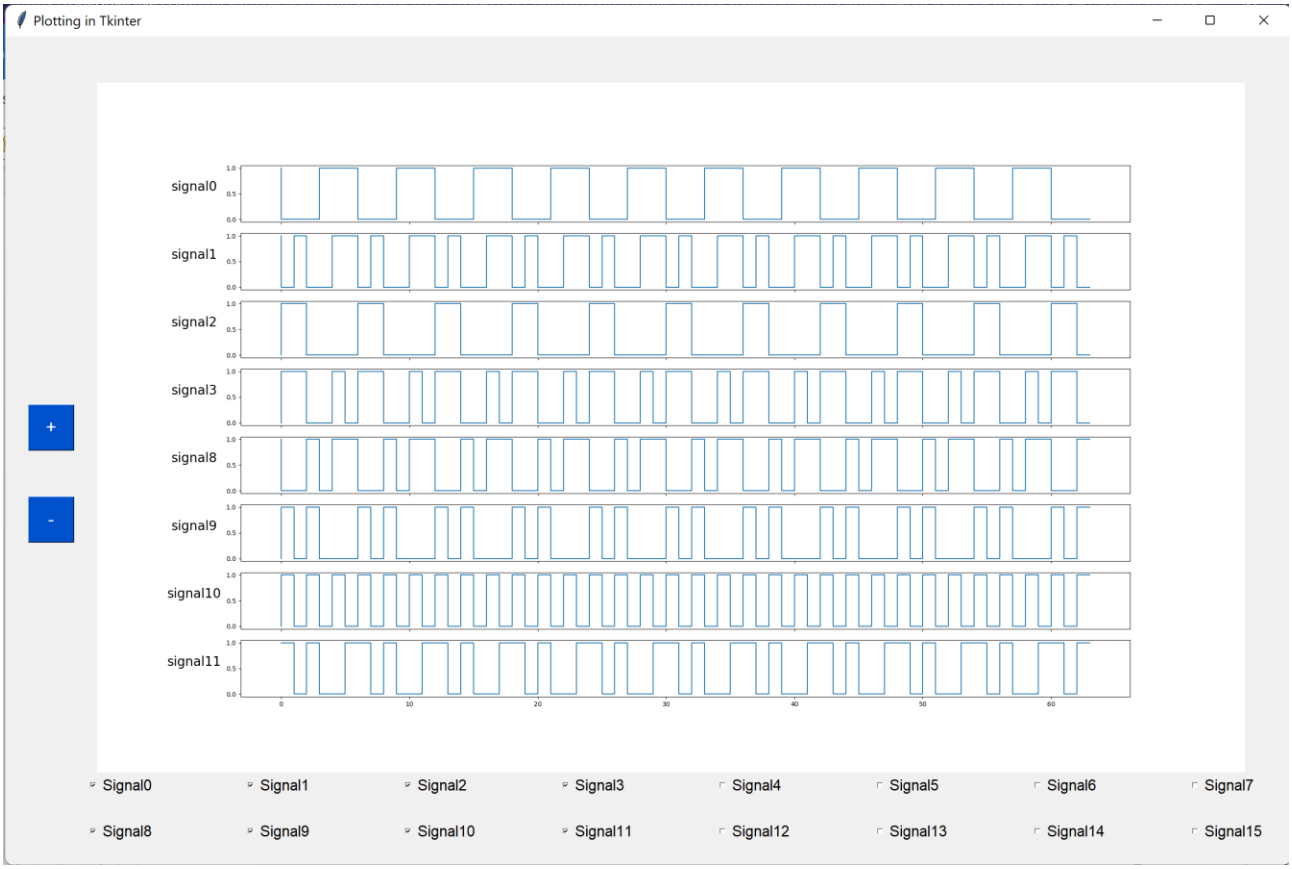
شکل ۲ نمودار سیگنال‌ها پس از چند عملیات Zooom in

نحوه کارکرد زوم بدین شکل است که ما سیگنال‌های ورودی آردوینو را تا میزان مشخصی ذخیره می‌کنیم و در هر لحظه تعداد مشخصی از مقادیر اخیر هر سیگنال را نمایش می‌دهیم و برای زوم کردن، با توجه به جهت زوم این تعداد را کاهش یا افزایش می‌دهیم. کد زیر نحوه پیاده‌سازی این قسمت را نشان می‌دهد:

```
def set_zoom_buttons():
    def zoom_in():
        global L
        if L >= MIN_L * 2:
            L = L // 2
            init_plot()
    def zoom_out():
        global L
```

```
        if L <= MAX_L // 2:
            L = L * 2
            init_plot()
    zoom_in_button = Button(window, text ="+", command = zoom_in, bg='#0052cc', f
g='#ffffff', font=font.Font(size=30))
    zoom_in_button.place(x=50, y=height // 2 - 100, height=100, width=100)
    zoom_out_button = Button(window, text ="-
", command = zoom_out, bg='#0052cc', fg='#ffffff', font=font.Font(size=30))
    zoom_out_button.place(x=50, y=height // 2 + 100, height=100, width=100)
```

در پایین صفحه نیز تیک‌های مربوط به هر سیگنال ورودی آردوینو را داریم که با برداشتن تیک مربوط به هر سیگنال نمودار آن حذف می‌شود. برای مثال با برداشتن تیک نیمی از سیگنال‌ها نمودار به شکل زیر در می‌آید:



شکل ۳ نمودار سیگنال‌ها پس از برداشتن تیک سیگنال‌های ۴، ۵، ۶، ۷، ۱۲، ۱۳، ۱۴ و ۱۵

نحوه پیاده‌سازی کلی این قسمت را در تکه کد زیر می‌توان دید:

```
def set_checkboxes():
```

```
    global shown_signals
    tmp_shown_signals = [IntVar(value=1) for i in range(N)]
    def change_checkbox():
        for i in range(N):
            shown_signals[i] = tmp_shown_signals[i].get()
        init_plot()

    for i in range(N):
        c1 = Checkbutton(window, text='Signal{}'.format(i), font=font.Font(size=2
5), variable=tmp_shown_signals[i], onvalue=1, offvalue=0, command=change_checkbox
)
        if i < N // 2:
            y = height - 200
        else :
            y = height - 100
        x = (i % (N // 2)) * (width - 400) / (N // 2 - 1) + 180
        c1.place(x=x, y=y)
```

در نهایت در سمت سخت‌افزاری کد مربوط به تولید سیگنال ورودی آردوینو که در فاز قبل نیز استفاده کردیم (با تغییر تاخیر آن به مقدار کمی) را در زیر داریم که سیگنال‌های ورودی توسط آن خوانده و ارسال می‌شوند:

```
1
2  int nums[16] = {7,22,24,26,28,30,32,36,38,40,42,44,46,48,50,52};
3  int counter;
4  int inputPins[16] = {7,22,24,26,28,30,32,36,38,40,42,44,46,48,50,52};
5  int outputPins[16];
6  unsigned int cur;
7  int len=6;
8
9  void setup(){
10     Serial.begin(9600);
11     while (!Serial) {
12         ; // wait for serial port to connect. Needed for native USB port only
13     }
14     for (int i=0; i < 16; i++){
15         pinMode(inputPins[i],INPUT);
16         outputPins[i] = inputPins[i] + 1;
17         pinMode(outputPins[i],OUTPUT);
18     }
19 }
20
21
22 void loop(){
23     cur = 0;
24     for (int i=0; i < 16; i++){
25         digitalWrite(outputPins[i], 1 & (nums[i] >> counter));
26         int val = digitalRead(inputPins[i]);
27         cur += (val << i);
28     }
29     delay(1000);
30     counter = (counter + 1) % len;
31     Serial.println(cur);
32 }
```

شکل ٤ کد اجرایی روی آردوینو برای دریافت سیگنال‌ها و ارسال آن به کامپیوتر

در قسمت نرم‌افزاری نیز، ابتدا با کدی مشابه فاز قبلی و با استفاده از کتابخانه pyserial سیگنال‌ها را دریافت می‌کنیم و سپس نمایش نمودارها را به‌روز می‌کنیم که این قسمت از پیاده‌سازی در تکه کد زیر آمده است:

```
def update_signals():
    global signal_array, thread_lock
    ser = serial.Serial('COM5', 9600, timeout=1)
    while True:
        line = ser.readline()
        serial_signal = "{:016b}".format((int(line.decode('utf-
8').strip("\r\n"))))
        thread_lock.acquire()
        x = [i for i in range(L)]
        for i in range(N):
```

```
            signal_array[i] += [int(serial_signal[15 - i])]
            if len(signal_array[i]) > MAX_L:
                signal_array = signal_array[-MAX_L:]

        ind = 0
        for i in range(N):
            if not shown_signals[i]:
                continue
            axs_step[ind].set_xdata(x)
            axs_step[ind].set_ydata(signal_array[i][-L:])
            plt.draw()
            ind += 1
        thread_lock.release()
        time.sleep(1)
```