



آزمایشگاه سخت افزار

گزارش پایانی پروژه
لاجیک آنالیزر و تستر دیجیتال

استاد: دکتر اجلاّلی

تیم شماره ۷ - اعضای تیم:

مهرداد صابری - ۹۷۱۱۰۱۳۳

محمد مهدوی - ۹۷۱۱۰۲۲۸

مسیح اسکندر - ۹۷۱۰۵۷۳۶

فهرست

- ۱ مقدمه و معرفی محصول ۳
- ۲ ساختار پیاده‌سازی محصول ۴
- ۳ قطعات استفاده شده ۵
- ۴ برنامه آردوینو ۵
- ۵ برنامه پایتون لاجیک آنالایزر ۸
- ۶ برنامه پایتون تستر دیجیتال ۱۶
- ۷ جمع‌بندی ۲۰

۱ مقدمه و معرفی محصول

در طول این پروژه، ما یک لاجیک آنالایزر نرم‌افزاری و تستر دیجیتال را طراحی کردیم و آن را توسعه دادیم. این محصول مطابق اسمش از دو ماژول اصلی لاجیک آنالایزر و تستر دیجیتال تشکیل شده است.

در ماژول لاجیک آنالایزر نرم‌افزاری، ما سیستمی مبتنی بر یک برد آردوینو را توسعه داده‌ایم که می‌تواند ۱۶ ورودی سیگنال دیجیتال را در قسمت سخت‌افزاری دریافت کند، سیگنال‌ها را به یک رایانه متصل به قسمت سخت‌افزاری از طریق پورت سریال ارسال کند و نمودار آن‌ها را در بستری نرم‌افزاری زنده نمایش دهد. در نرم‌افزار طراحی شده، ۱۶ سیگنال در یک رابط گرافیکی همزمان به کاربر نمایش داده می‌شوند، و قابلیت‌های مختلفی وجود دارند که به کاربر اجازه تغییر نمایش سیگنال‌ها را می‌دهد. اولاً، کاربر می‌تواند در رابط گرافیکی مشخص کند که کدام سیگنال‌ها را می‌خواهد نمایش دهد و هر سیگنال را به صورت جدا پنهان یا نمایان کند. یک ویژگی دیگر نرم‌افزار این است که در آن کاربر می‌تواند سطح زوم روی نمودارها را تغییر دهد و روی نمودارها Zoom یا Zoom in کند. در کنار این‌ها، نرم‌افزار ما این قابلیت را دارد که هر سیگنال ورودی را در یک فایل روی رایانه ذخیره کند و می‌تواند سیگنال‌های ذخیره شده از قبل را لود کند و دوباره نمایش دهد. نرم‌افزار شامل قابلیت‌های تزئینی‌ای مثل تغییر رنگ نمودارها نیز هست.

ماژول تستر دیجیتال، قسمت دیگر این محصول است که در آن کاربر می‌تواند با استفاده از همان رابط کاربری ماژول آنالایزر، سیگنال‌های دلخواه خود را تعیین کند تا در قسمت سخت‌افزاری به عنوان سیگنال خروجی خروجی داده شوند. برای ایجاد امکان تست IC‌های با دو ورودی، سیستم ما از تعیین دو سیگنال خروجی پشتیبانی می‌کند.

مشخصات این سیستم به صورت کلی به صورت زیر است:

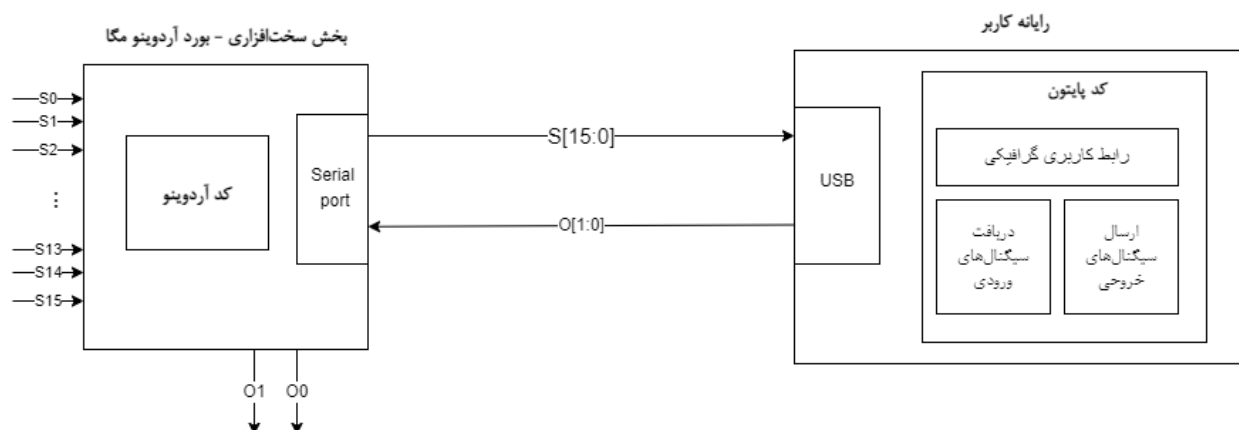
ATmega2560	میکروکنترلر
۵ ولت	ولتاژ کاری
۷-۱۲ ولت	ولتاژ ورودی پیشنهادی
۱۶	تعداد پین‌های دیجیتال ورودی
۲	تعداد پین‌های دیجیتال خروجی
۲۰ میلی‌آمپر	جریان پین‌های ورودی و خروجی
۱۰ سانتی‌متر	طول
۵/۵ سانتی‌متر	عرض
۳۷ گرم	وزن
کابل USB	نحوه اتصال

در ادامه‌ی این گزارش به بخش‌های زیر خواهیم پرداخت:

- مقدمه‌ی فنی شامل توضیح ساختار کلی نرم‌افزار، کتابخانه‌ها و قطعات مورد استفاده
- توضیح کد مربوط به بخش آردوینو
- توضیحات لاجیک آنالایزر شامل بخش‌های مختلف این ابزار به همراه عکس از محیط نرم‌افزار
- توضیحات تستر دیجیتال شامل نحوه‌ی کارکرد بخش‌های گرفتن سیگنال در محیط گرافیکی و ارسال آن به آردوینو
- جمع‌بندی نهایی شامل ویدئوی تست‌های انجام‌شده، مقایسه تخمین اولیه قیمت محصول و قیمت نهایی آن، و توضیحات بسته‌بندی محصول

۲ ساختار پیاده‌سازی محصول

ساختار محصول آنالایزر و تستر، به صورت کلی از دو بخش سخت‌افزاری و نرم‌افزاری تشکیل شده است. قسمت سخت‌افزاری سیستم، به طور کلی از یک برد آردوینو مگا تشکیل شده است که از پین‌های ورودی و خروجی دیجیتال آن استفاده می‌شود تا سیگنال‌های ورودی کاربر دریافت و سیگنال‌های خروجی به او تحویل داده شود. در سمت نرم‌افزاری، قسمت عمده سیستم را برنامه سمت رایانه کاربر تشکیل می‌دهد، که با دریافت سیگنال‌ها از آردوینو آن‌ها را در رابط کاربری گرافیکی خود نمایش می‌دهد، و با استفاده از همین رابط کاربری خروجی‌های مورد نظر کاربر را از او می‌گیرد و به سخت‌افزار ارسال می‌کند. علاوه بر این برنامه، برنامه‌ای برای کنترل برد آردوینو نوشته شده است که امکان دریافت ورودی و دادن خروجی را برقرار می‌کند. در شکل ۱ شمای کلی این سیستم به صورت تصویری آمده است که اجزای مختلف سیستم و ارتباطات آن‌ها را نشان می‌دهد. در مخزن کد مربوط به این پروژه، فایل برنامه سمت کاربر با نام `tester-analyzer.py` و فایل برنامه سمت آردوینو با نام `arduino-io-code.ino` هر دو در پوشه مربوط به کد نهایی آمده‌اند.



شکل ۱ شمای کلی ارتباطات و اجزای سیستم پیاده‌سازی شده

برای پیاده‌سازی برنامه سمت کاربر، از زبان برنامه‌نویسی پایتون (Python) استفاده شده است و قسمت‌های مختلف سیستم به صورت تابع‌های مختلف پیاده‌سازی شده‌اند. در این برنامه، از چند کتابخانه آماده پایتون برای کمک به پیاده‌سازی استفاده شده است. یکی از کتابخانه‌های استفاده شده، کتابخانه `serial` (یا `pyserial`) است که امکان ارتباط از طریق پورت سریال با دستگاه آردوینو را برقرار می‌کند. یک کتابخانه دیگر که در ساخت این برنامه استفاده شده است، کتابخانه `matplotlib` است و قسمت `pyplot` آن است. این کتابخانه با توجه به برقرار کردن امکان نمایش نمودارهای متنوع، برای نمایش نمودارهای سیگنال‌های ورودی دریافتی و سیگنال‌های خروجی ارسالی استفاده می‌شود. در آخر، از کتابخانه `tkinter` برای ساختن رابط کاربری گرافیکی نرم‌افزار استفاده شده است.

در ساختار کد، دو قسمت اصلی سیستم، یعنی بخش آنالایزر و تستر، در کنار هم و با استفاده از المان‌های مختلف یک رابط کاربری گرافیکی پیاده‌سازی شده‌اند، و قسمتی از کد به صورت مشترک مربوط به این دو کارکرد متفاوت است. با وجود اشتراکات کد، در ادامه این دو بخش را به صورت جدا از هم بررسی می‌کنیم.

در ادامه، هر یک از بخش‌های تشکیل‌دهنده سیستم را با جزئیات بیشتر بررسی می‌کنیم.

۳ قطعات استفاده شده

همانطور که در بخش قبلی گفته شد، برای پیاده‌سازی سیستم از یک قطعه آردوینو مگا استفاده می‌کنیم. شکل یک نمونه از این قطعه در شکل ۲ آمده است. در شکل ۳ هم تصویر سیم رابط آمده است که برای اتصال آردوینو به رایانه استفاده می‌شود.



شکل ۲ نمونه یک قطعه آردوینو مگا



شکل ۳ سیم رابط آردوینو و رایانه

علاوه بر این قطعات، از سیم برای اتصال قطعات دیگر به پین‌های خروجی و ورودی قطعه آردوینو در محصول استفاده می‌شود.

۴ برنامه آردوینو

همانطور که در بخش‌های پیشین گفته شد، در سیستم از یک آردوینو مگا برای دریافت ورودی‌ها و خروجی دادن خروجی‌ها استفاده می‌شود که عملکرد این قطعه توسط یک برنامه که روی آن اجرا می‌شود کنترل می‌شود. این کد از دو بخش شروع کار و حلقه اجرایی اصلی تشکیل شده است که در ادامه این بخش‌ها را بررسی می‌کنیم.

```

int inputPins[16] = {0, 1, 2, 3, 4, 5, 6, 7, 14, 15, 16, 17, 18, 19, 20, 21};
int singal_out_pins[2] = {12, 13};
int incoming_byte;
unsigned int cur;

void setup(){
    // start serial connection with baud rate 9600
    Serial.begin(9600);
    while (!Serial) {
        ; // wait for serial port to connect. Needed for native USB port only
    }
    // declare input pins
    for (int i=0; i < 16; i++){
        pinMode(inputPins[i],INPUT);
    }
    // decalre output pins
    pinMode(singal_out_pins[0], OUTPUT);
    pinMode(singal_out_pins[1], OUTPUT);
}

```

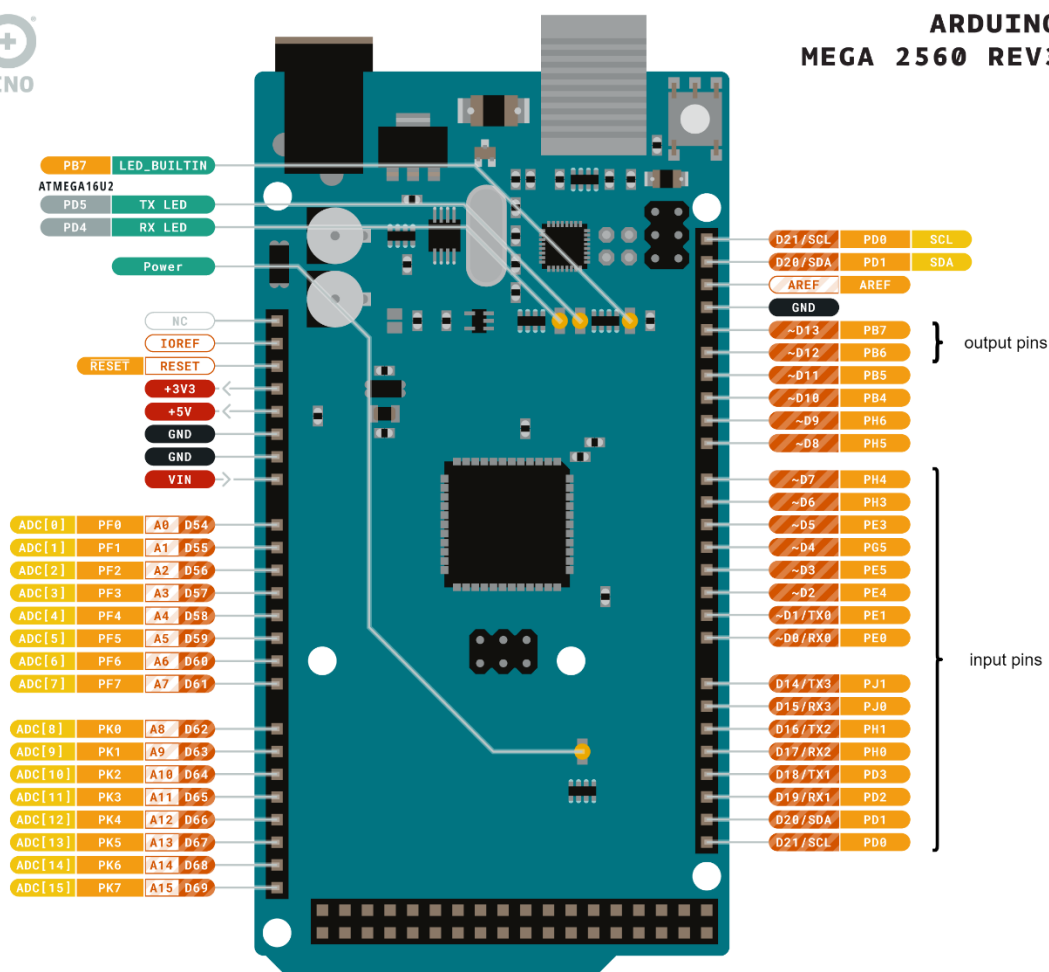
شکل ۴ کد مربوط به شروع کار قطعه آردوینو

شکل ۴ قطعه‌ای از کد را نشان می‌دهد که مربوط به شروع کار قطعه آردوینو است. همچنین متغیرهای استفاده شده در برنامه در این قسمت تعریف می‌شود. در ابتدای این قطعه کد، تعریف شماره پین‌های دیجیتال آردوینو را داریم که به عنوان پین خروجی یا ورودی در سیستم ما استفاده می‌شوند. در شکل ۵ محل قرارگیری پین‌های ورودی و خروجی سیستم آنالایزر و تستر ما بر روی نمای pinout نمایش داده شده است. همانطور که در شکل و کد نشان داده شده است و در مشخصات سیستم گفته شده، از ۱۶ پین دیجیتال آردوینو به عنوان پین ورودی و از ۲ پین دیجیتال به عنوان خروجی استفاده می‌کنیم.

پس از تعریف متغیرهای برنامه در این قطعه کد، قسمت setup را داریم که راه‌اندازی قطعه در این قسمت انجام می‌شود. در این قسمت ارتباط با رایانه از سمت این برنامه با استفاده از Serial برقرار می‌شود. در برقراری ارتباط، عدد ۹۶۰۰ به عنوان نرخ baud مشخص شده است که در برنامه سمت کاربر نیز همین مقدار استفاده شده است. در پایان این قسمت از کد، مد کاری پین‌های مورد استفاده به عنوان خروجی یا ورودی مشخص شده است.



ARDUINO MEGA 2560 REV3



شکل ۵ پین اوت قطعه آردوینو مگا به همراه نشانگر محل قرارگیری پین های ورودی و خروجی سیستم

قسمت دیگر کد آردوینو، قسمت loop آن است که پس از آغاز کار قطعه به صورت مداوم در قطعه اجرا می شود. این قطعه از کد در شکل ۶ نمایش داده شده است. در ابتدای این قطعه کد، در یک حلقه، ۱۶ مقدار تکبیتی مربوط به سیگنال های ورودی در لحظه فعلی از پین های مربوطه دریافت می شود و به عنوان یک عدد ۱۶ بیتی روی پورت سریال فرستاده می شود. برای این کار، مقدار هر یک از پین های مشخص شده به عنوان ورودی در قسمت قبلی کد، توسط فرمان `digitalRead` خوانده می شود و مقدار یکی از بیت های اول تا شانزدهم عدد فرستاده شده را مشخص می کند.

در ادامه این قطعه کد، کد مربوط به دریافت مقادیر از پورت سریال و قرار دادن این مقادیر در پین های خروجی را داریم. در این قسمت، یک بایت از پورت سریال دریافت می شود. مقدار این بایت با توجه به کد سمت رایانه کاربر، کاراکتر `ASCII` مربوط به یکی از ارقام ۰ تا ۳ است و مقدار سیگنال های خروجی بر اساس بیت های این رقم مشخص می شود. برای تبدیل این مقدار `ASCII` به عددی بین ۰ تا ۳، مقدار ۴۸ که کد `ASCII` رقم ۰ است را از آن کم می کنیم و در نهایت بیت های مورد نظر را با دستور `digitalWrite` در پین های خروجی قرار می دهیم.

```

void loop(){
  // read input signals and encode them to a 16 bit value
  cur = 0;
  for (int i=0; i < 16; i++){
    int val = digitalRead(inputPins[i]);
    cur += (val << i);
  }
  // send encoded input signals using serial port
  Serial.println(cur);
  delay(1000);
  if (Serial.available()){
    // receive outputs from serial port and decode them
    incoming_byte = Serial.read();
    incoming_byte = incoming_byte - 48;
    // write output values to output pins
    for (int i=0; i < 2; i++){
      digitalWrite(singal_out_pins[i], 1 & (incoming_byte >> i));
    }
  }
  delay(1000);
}

```

شکل ۶ قطعه کد مربوط به حلقه اصلی اجرایی در قطعه آردوینو

با توجه به این برنامه، قطعه آردوینو پس از شروع به کار، به طور مداوم مقادیر ورودی خود را از پورت سریال به رایانه ارسال می‌کند و همچنین مقادیر ارسال شده توسط رایانه را نیز دریافت می‌کند و در پین‌های مشخص شده به عنوان خروجی قرار می‌دهد.

۵ برنامه پایتون لاجیک آنالایزر

در این فصل بخش‌های مختلف ماژول لاجیک آنالایزر را شرح می‌دهیم.

۵/۱ دریافت ورودی از آردوینو

گرفتن ورودی‌ها از آردوینو توسط کتابخانه Pyserial در Python انجام می‌شود. در این سیستم ۱۶ سیگنال ورودی داریم که در بازه‌ی زمانی مشخصی که همان کلاک سیستم است مقدارشان از آردوینو گرفته شده و با کمک پورت سریال (USB) به نرم‌افزار داده می‌شود.

نمای کلی بخش دریافت ورودی در کد مطابق شکل ۷ است که در آن سیگنال‌ها پس از دریافت دیکد و سپس مورد استفاده نرم‌افزار قرار می‌گیرند.

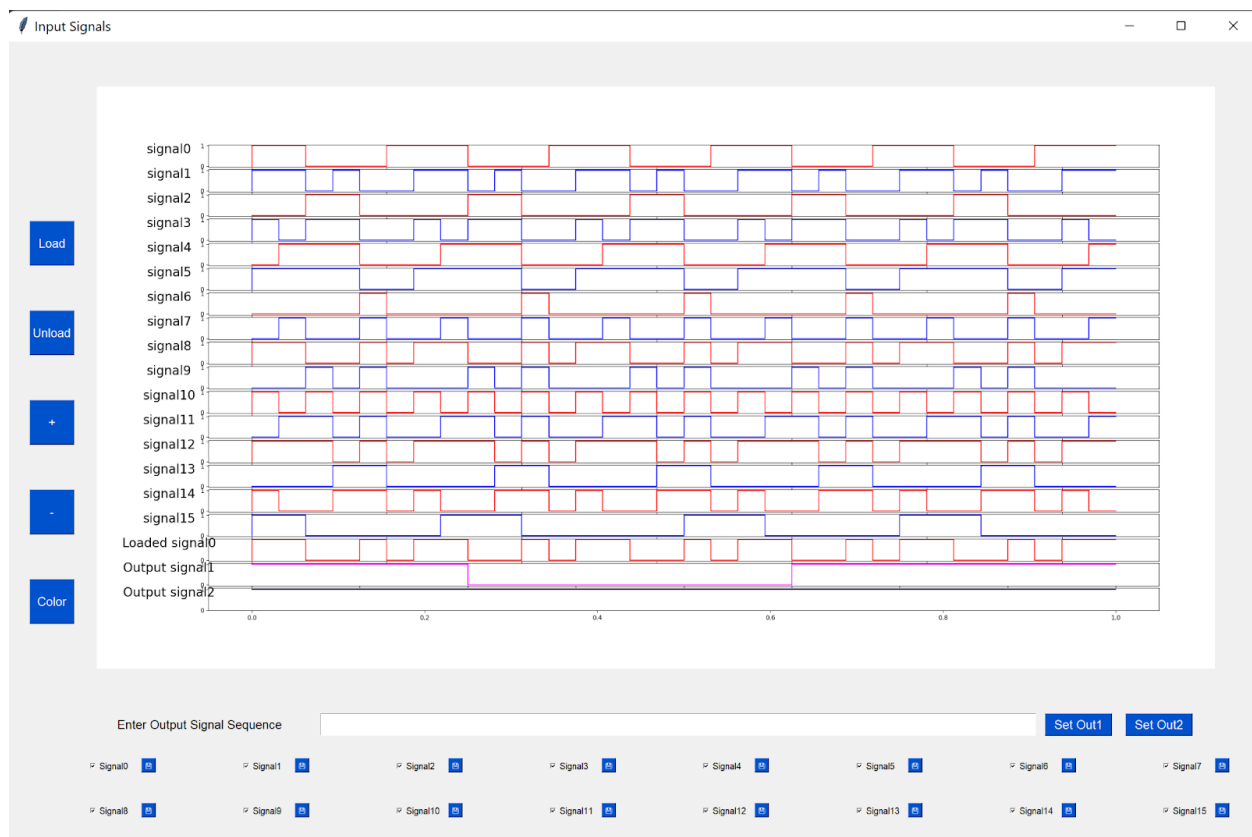

```

ser = serial.Serial(serial_port, 9600, timeout=1)
while True:
    line = ser.readline()
    # Decode the signals ...
    time.sleep(1)
    # Process the signals
    time.sleep(1)

```

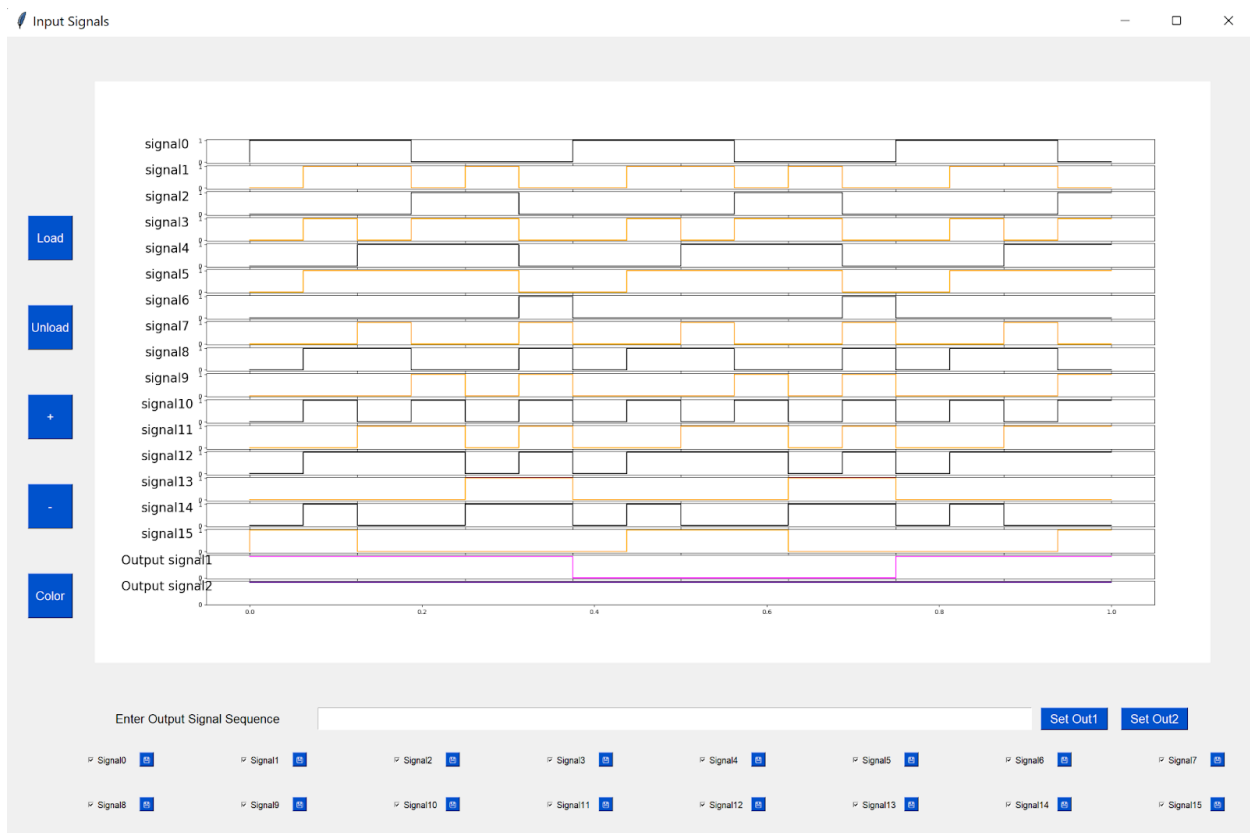
شکل ۷ قسمت دریافت ورودی در کد پایتون

۵/۲ نمایش سیگنال‌ها



شکل ۸ نمایش سیگنال‌ها

با استفاده از کتابخانه tkinter در Python یک رابط کاربری برای نمایش سیگنال‌ها پیاده سازی کردیم و در این رابط کاربری با استفاده از کتابخانه Matplotlib سیگنال‌ها را رسم کردیم. همانطور که در شکل ۸ قابل مشاهده است، در محیط گرافیکی نرم‌افزار می‌توانیم ۱۶ سیگنال ورودی (با نام‌های signal0, signal1, ..., signal15) 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.



شکل ۹ تم‌های رنگی متفاوت نمودارها

در نمایش سیگنال‌ها از یک دکمه‌ی Color استفاده کرده‌ایم که با کلیک روی آن تم رنگی نمایش سیگنال‌ها تغییر می‌کند. رنگ‌های سیگنال‌ها به صورت یکی در میان متفاوت گذاشته شده‌است تا بتوان آنها را راحت‌تر از هم تمایز داد. یک نمونه از تم رنگی متفاوت با تم اولیه در شکل ۹ آمده‌است.

برای اینکه بتوانیم نمایش سیگنال‌ها به صورت زنده را با کمک کتابخانه Matplotlib انجام دهیم از توابع `set_ydata` و `set_xdata` هر بار که مقدار جدیدی به سیگنال‌ها اضافه می‌شود استفاده کردیم. اما در برخی موارد که نیاز بود تغییری در نمایش سیگنال‌ها داده‌شود نیاز بود که نمودارها از نو رسم شوند. برای مثال برای تغییر رنگ یا قابلیت‌هایی که در ادامه آنها را بررسی خواهیم کرد نظیر ذخیره‌سازی و بازیابی سیگنال‌ها این کار انجام شده‌است.

۵/۳ زوم کردن

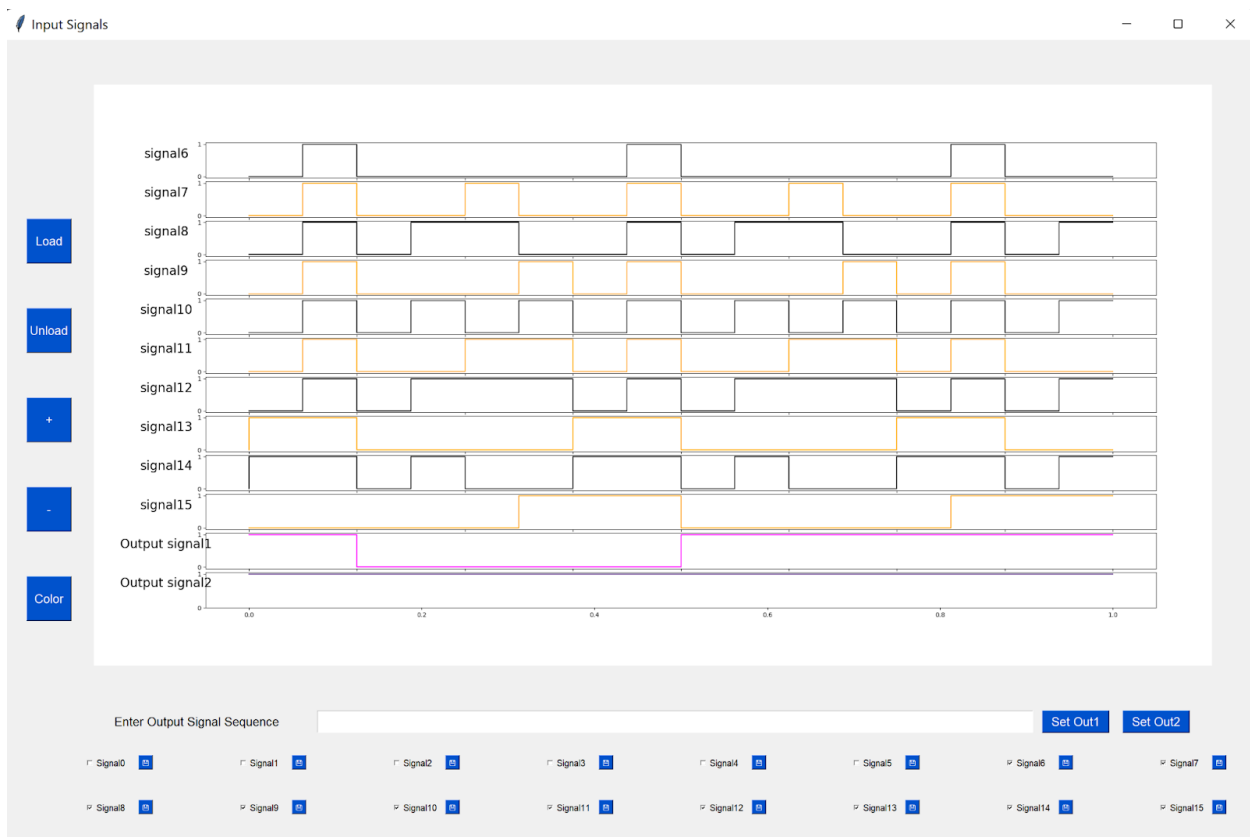


شکل ۱۰ نمودارها در حالت Zoom in شده

همانطور که در شکل ۸ قابل مشاهده است، دو دکمه با لیبل های + و - در سمت چپ محیط کاربری داریم که کارکرد آنها Zoom in و Zoom out کردن نمایش نمودار سیگنال ها است. با هر بار کلیک کردن روی این دکمه ها تعداد مقادیر نمایش داده شده برای هر سیگنال دو برابر کم یا زیاد می شود. در نرم افزار قابلیت Zoom out کردن تا نمایش ۱۰ مقدار، و Zoom in کردن تا نمایش ۱ مقدار برای هر سیگنال گذاشته شده است. در شکل ۱۰ نمونه ای از محیط نرم افزار که Zoom out شده است قرار داده شده است و شکل ۲ نیز حالت یک قدم بیشتر Zoom in شده در مقایسه به شکل ۸ است. پیاده سازی این کاربرد به این صورت است که برای هر سیگنال یک آرایه از ۱۰۲۴ مقدار آخرش نگهداری می شود اما فقط تعداد مشخصی از این مقادیر بر حسب مقدار زوم خواسته شده توسط کاربر نمایش داده می شوند. در ازای هر بار کلیک روی این دو دکمه، نمودارها از نو با کتابخانه Matplotlib ساخته می شوند.

۵/۴ حذف و انتخاب سیگنال ها

علاوه بر قابلیت نمایش تمام ۱۶ سیگنال ورودی در محیط نرم افزاری، قابلیت انتخاب برخی از سیگنال ها و عدم نمایش بقیه سیگنال ها موجود است. همانطور که در شکل ۸ می بینید، اینکار با استفاده از Checkbox های مربوط به هر سیگنال ورودی در بخش پایین محیط کاربری انجام می شود. همانند قابلیت زوم کردن، در اینجا هم پس از هر تغییر در سیگنال های انتخابی برای نمایش، نمودارها از نو با کتابخانه Matplotlib ساخته می شوند.



شکل ۱۱ انتخاب سیگنال‌ها برای نمایش

در شکل ۱۱ یک نمونه از محیط نرم‌افزار در حالتی که تمامی سیگنال‌ها برای نمایش انتخاب نشده‌اند آمده‌است.

۵/۴ ذخیره و بازیابی سیگنال‌ها

```
def set_save_buttons():
    # show fileselect dialog and save the chosen signal in the selected file
    # saves each signal value on a seperate line
    class button_command_obj:
        def __init__(self, ind) -> None:
            self.ind = ind
        def save_signal(self):
            nums = signal_array[self.ind][:]
            print(nums)
            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()
    # set up save buttons in the GUI
    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=button_command_obj(i).save_signal,
            bg='#0052cc', fg='ffffff', font=font.Font(size=12))
        btn.place(x=x, y=y)
```

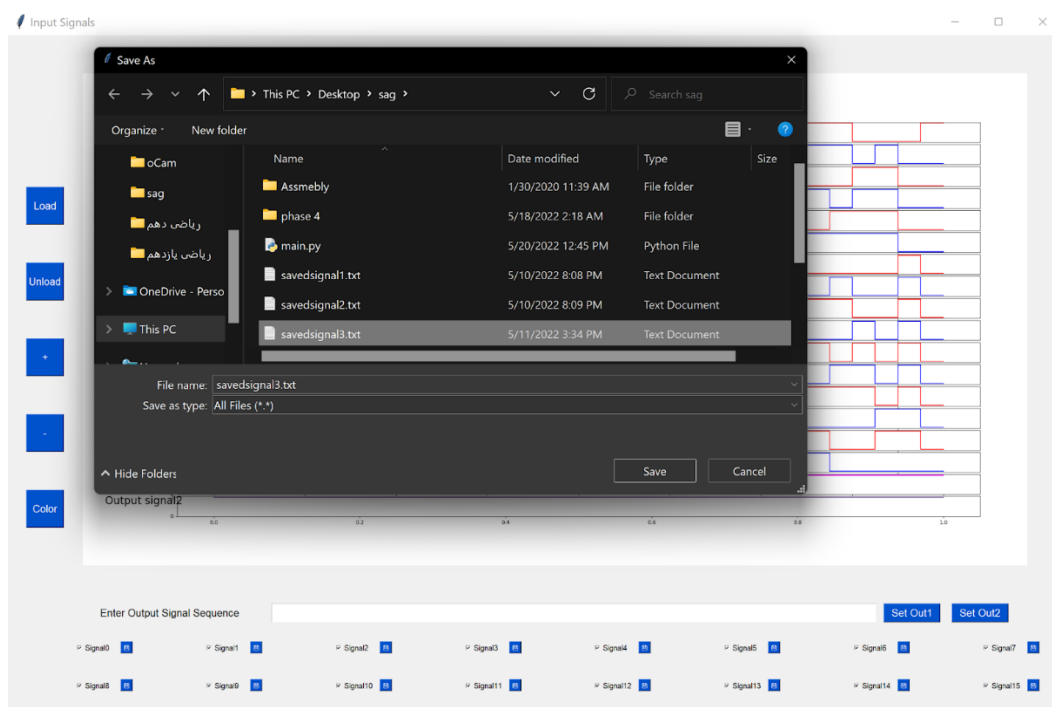
شکل ۱۲ کد مربوط به ذخیره سیگنال‌ها

تکه کد موجود در شکل ۱۲ 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 لحظه فشرده شدن دکمه ذخیره می‌شوند و بعداً می‌توانیم این مقادیر را بازیابی کنیم.

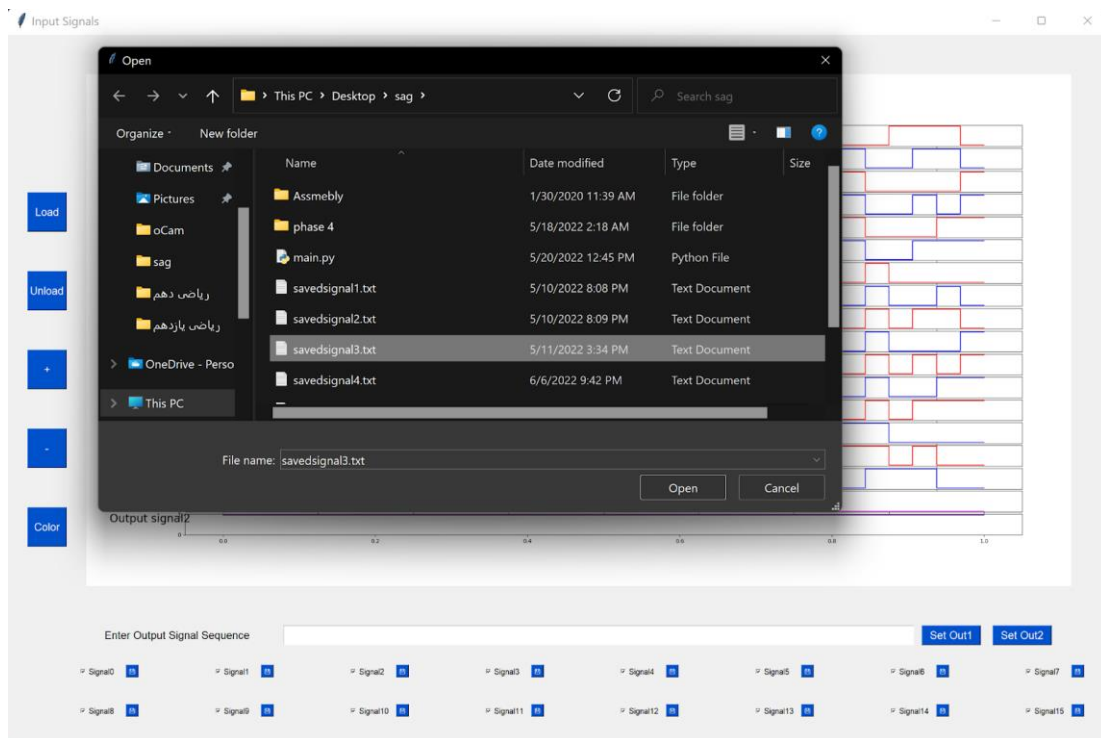
همانطور که در شکل ۸ مشخص است، دو دکمه‌ی `Load` و `Unload` نیز داریم. `Load` برای بازیابی یک فایل و نمایش سیگنال‌ش، و `Unload` برای پاک کردن آخرین سیگنال بازیابی شده از لیست سیگنال‌های در حال نمایش را می‌سازد. هرکدام از این دکمه‌ها در صورت کلیک شدن تابع مخصوصشان را صدا می‌زنند. این دو تابع در تکه کد شکل ۱۳ آورده شده‌اند:



شکل ۱۳ کد مربوط به قابلیت لود سیگنال‌ها

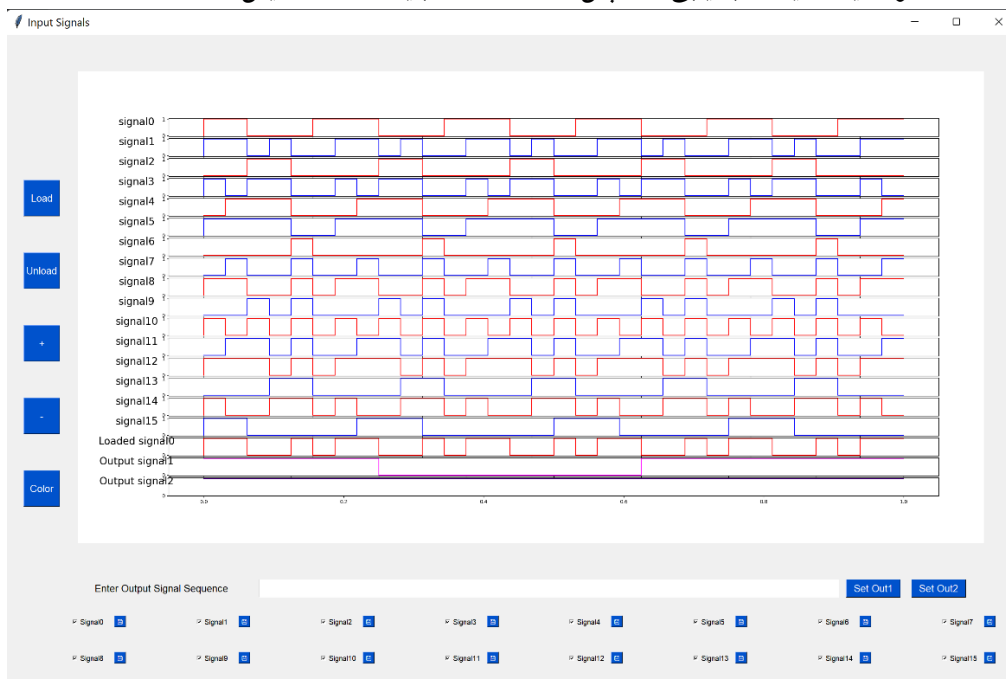


شکل ۱۴ صفحه انتخاب مکان ذخیره سیگنال



شکل ۱۵ صفحه انتخاب فایل برای بازیابی سیگنال ذخیره شده

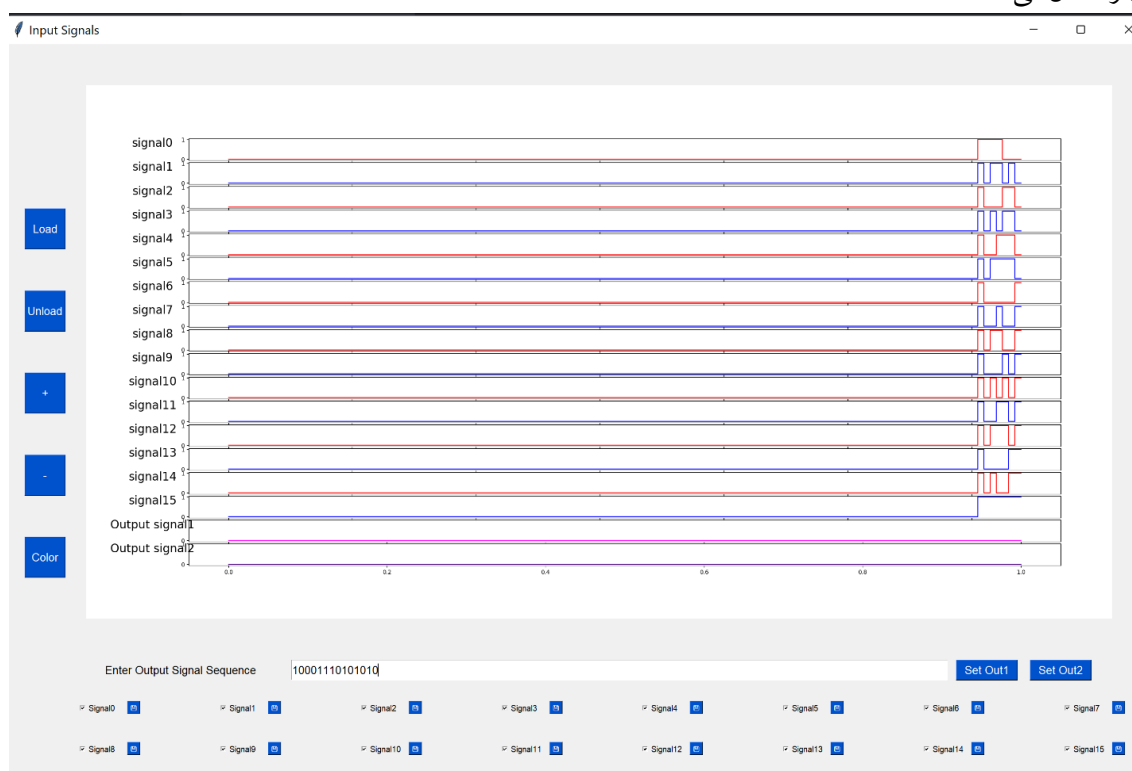
با کلیک روی دکمه‌ی Save و Load مانند شکل ۱۴ و شکل ۱۵ محیطی برای انتخاب مسیر ذخیره‌سازی و فایل جهت بازیابی باز می‌شود. در شکل ۱۶ نمودار یک سیگنال بازیابی شده پس از استفاده از قابلیت Load نمایش داده شده است.



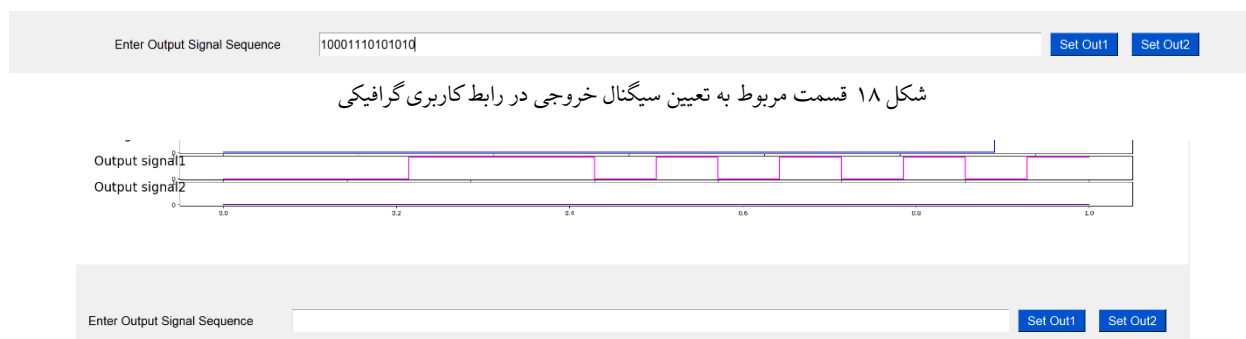
شکل ۱۶ نمودارها پس از بازیابی یک سیگنال ذخیره شده

۶ برنامه پایتون تستر دیجیتال

در این قسمت از برنامه، کد مربوط به دریافت سیگنال‌های خروجی از کاربر در رابط کاربری گرافیکی و ارسال آن‌ها از طریق پورت سریال به قطعه آردوینو را داریم. در شکل ۱۷ و شکل ۱۸ قسمت‌های مربوط به این کاربرد سیستم در رابط کاربری گرافیکی مشخص شده است. همانطور که در تصاویر مشخص است، کاربر برای مشخص کردن یک سیگنال خروجی، می‌تواند دنباله ۰ و ۱ مربوط به آن را در تکست باکس مربوطه بنویسد و سیگنال متناظر با دنباله را با استفاده از یکی از دکمه‌ها به یکی از دو خروجی اختصاص دهد. لازم به ذکر است که با توجه به اجرای مداوم برنامه، سیگنال تعیین شده به صورت دوری و متناوب در خروجی قرار می‌گیرد. در شکل ۱۹ که نتیجه تخصیص سیگنال نوشته شده در شکل ۱۷ به سیگنال خروجی اول است، قسمتی از رابط کاربری را می‌بینیم که مقادیر سیگنال‌های خروجی را به صورت نمودار نشان می‌دهد. این نمودار در طول زمان تغییر می‌کند تا در سمت راست آخرین خروجی ارسال شده را نشان دهد. لازم به ذکر است که نمودارهای سیگنال‌های خروجی با تغییر میزان زوم تغییر نمی‌کنند و همه‌ی سیگنال را نشان می‌دهند.



شکل ۱۷ شمای کلی رابط کاربری گرافیکی – مقداری در قسمت تعیین مقدار سیگنال خروجی نوشته شده است



شکل ۱۸ قسمت مربوط به تعیین سیگنال خروجی در رابط کاربری گرافیکی

شکل ۱۹ نتیجه اعمال تعیین سیگنال خروجی

چند بخش از کد پایتون پیاده‌سازی شده مربوط به پیاده‌سازی این بخش از سیستم هستند که در ادامه توضیحات آن‌ها را داریم. شکل ۲۰ متغیرهای مربوط به سیگنال‌های خروجی در برنامه را نشان می‌دهد. در ابتدا، متغیر `output_signals` را داریم که حاوی مقادیر سیگنال‌های خروجی است که قرار است به قطعه آردوینو ارسال شوند. در طول ارسال یک سیگنال، این مقادیر تغییری نمی‌کنند و مقدار `output_inds` است که تعیین می‌کند بیت لحظه چندم سیگنال ارسال شود.

```
# signals sent to output
output_signals = [[0], [0]]

# indices for cycling through output signals
output_inds = [0, 0]
```

شکل ۲۰ متغیرهای مربوط به سیگنال‌های خروجی

در شکل ۲۱ قطعه کد مربوط به تنظیم و قرار دادن المان‌های گرافیکی مربوط به تعیین سیگنال‌های خروجی در رابط کاربری آمده است. در این قطعه کد، تکست باکس ورود دنباله سیگنال، دکمه‌های تخصیص و یک لیبل برای مشخص کردن وظیفه تکست باکس ساخته می‌شوند و در محیط گرافیکی قرار می‌گیرند. می‌توان دید که مقدار وارد شده در تکست باکس، با فشردن هر کدام از دکمه‌های تخصیص به سیگنال خروجی مربوطه اختصاص داده می‌شود و نمودارها دوباره با مقدار جدید سیگنال‌ها کشیده می‌شوند. همچنین، مقدار اندیس شمارنده سیگنال خروجی مربوطه (`output_inds`) به مقدار ۰ برگردانده می‌شود تا برنامه ارسال این سیگنال را از ابتدای آن شروع کند.

```

# set up textboxes and buttons to input signals to be sent to arduino
def set_output_textbox():
    # set output signal based on string sequence of 0s and 1s
    def submit_output(index):
        global output_signals, output_inds
        out_text = text_var.get()
        text_var.set("")
        if not out_text or not re.match("[01]*", out_text):
            return
        output_signals[index] = [int(c) for c in out_text]
        output_inds[index] = 0
        init_plot()

    # set up the elements in the GUI
    output_label = Label(window, text="Enter Output Signal Sequence", font=font.Font(size=20))
    text_var = StringVar()
    output_textbox = Entry(window, textvariable=text_var, font=font.Font(size=20))
    sub_btn1 = Button(window, text = "Set Out1", command = lambda : submit_output(0),
font=font.Font(size=20), bg='#0052cc', fg="ffffff")
    sub_btn2 = Button(window, text = "Set Out2", command = lambda : submit_output(1),
font=font.Font(size=20), bg='#0052cc', fg="ffffff")
    output_label.place(x=180, y= height - 300, height=50, width=500)
    output_textbox.place(x=700, y= height - 300, height=50, width=1600)
    sub_btn1.place(x=2320, y=height - 300, height= 50, width=150)
    sub_btn2.place(x=2500, y=height - 300, height= 50, width=150)

```

شکل ۲۱ تکه‌کد راه‌اندازی المان‌های رابط کاربری گرافیکی مربوط به تعیین سیگنال‌های خروجی

در شکل ۲۲ قطعه کد مربوط به سیگنال‌های خروجی در تابع `init_plot` (که وظیفه کشیدن نمودار از اول را دارد) آمده است. در این قطعه کد، ابتدا رنگ‌های نمودار دو سیگنال خروجی مشخص شده اند که رنگ‌های صورتی و بنفش هستند. در ادامه، در درون حلقه برای هر سیگنال خروجی، مقادیر x و y برای رسم آن سیگنال مشخص شده است و نمودار پله‌ای آن به عنوان زیرنمودار نمودار اصلی اضافه می‌شود. در مشخص کردن مقدار y سیگنال رسم شده، از مقدار `output_inds` مربوط به سیگنال استفاده می‌شود تا آخرین مقدار ارسال شده سیگنال سمت راست‌ترین مقدار نمایش داده‌شده در نمودار آن باشد.

```
def init_plot():
    ...
    # output signal colors
    out_colors = ["fuchsia", "indigo"]
    for i in range(len(output_signals)):
        # plot all signal elements for outputs
        x_out = [j for j in range(len(output_signals[i]) + 1)]
        ydata = output_signals[i][output_inds[i]:] + output_signals[i][:output_inds[i]]
        axs_step += [axs[ind].step(x_out, [ydata[0]] + ydata)[0]]
        axs[ind].set_yticks([0,1])
        # set plot line color
        axs_step[ind].set_color(out_colors[i % 2])
        axs[ind].set_ylabel("Output signal{}".format(
            i + 1), rotation=0, fontsize=20, labelpad=50, horizontalalignment="center")
        ind += 1
    ...
```

شکل ۲۲ قطعه کد مربوط به سیگنال‌های خروجی در تابع از سر کشیدن نمودارها

در شکل ۲۳، تکه کدهای مربوط به سیگنال‌های خروجی در تابع `update_signals` را داریم. این تابع که شامل یک حلقه `while` بدون انتها است، حلقه اصلی برنامه را تشکیل می‌دهد که در آن ارتباط با قطعه آردوینو انجام می‌شود. در قطعه کد اول درون این حلقه که در شکل دیده می‌شود، ابتدا مقدار بیت‌های فعلی سیگنال‌های خروجی (که با `output_inds` مشخص شده است) به عنوان یک عدد ۲ بیتی در نظر گرفته می‌شود و سپس به عنوان یک رشته تک‌کاراکتری روی پورت سریال ارسال می‌شود. در قسمت کد آردوینو دیدیم که این مقدار در طرف دیگر به عنوان دریافت می‌شود و با استفاده از خواص ASCII به مقدار ۲ بیتی اصلی برگردانده می‌شود. در قسمت بعدی کد که در شکل آمده است، مقدار `y` نمودارهای سیگنال‌های خروجی با توجه به آپدیت شدن `output_inds` آپدیت می‌شود.

```
def update_signals():
    ...
    ser = serial.Serial(serial_port, 9600, timeout=1)
    while True:
        ...
        num_to_write = str(sum((2**i)*output_signals[i][output_inds[i]] for i in
range(len(output_signals))))
        for i in range(len(output_inds)):
            output_inds[i] = (output_inds[i] + 1) % len(output_signals[i])
        ser.write(num_to_write.encode())
        ...
        # update output signal plots
        for i in range(len(output_signals)):
            ydata = output_signals[i][output_inds[i]:] + output_signals[i][:output_inds[i]]
            axs_step[ind].set_ydata([ydata[0]] + ydata)
            ind += 1
        ...
```

شکل ۲۳ قسمت مربوط به سیگنال‌های خروجی در حلقه اصلی برنامه

۷ جمع بندی

در ادامه جمع بندی ای بر پروژه توسعه محصول داریم.

۷/۱ هزینه محصول

تخمین هزینه اولیه برای محصول در پروپوزال پروژه ۵۶۳,۰۰۰ تومان بوده است که این تخمین با قیمت بردبرد بوده است که فقط برای تست محصول استفاده شده و در محصول نهایی وجود ندارد. اما جدای از آن، بقیه هزینه های محصول نهایی مطابق تخمین اولیه بوده (فقط قیمت بسته بندی ۷,۰۰۰ تومان افزایش داشته است) و در کل قیمت خام (بدون در نظر گرفتن دستمزد ساخت) نهایی محصول شامل آردوینو مگا، سیم رابط USB، سیم، و بسته بندی برابر با ۵۳۷,۰۰۰ تومان می باشد.

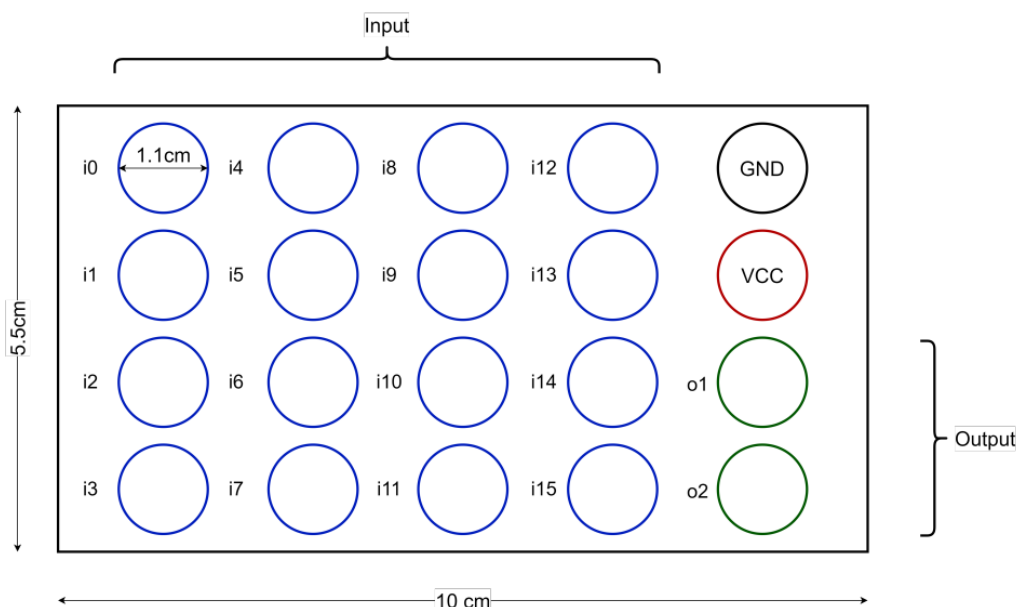
۷/۲ تست محصول

فرایند تست کردن قابلیت های محصول به صورت یک ویدئوی ضبط شده در [لینک](#) قرار داده شده است.

۷/۳ بسته بندی

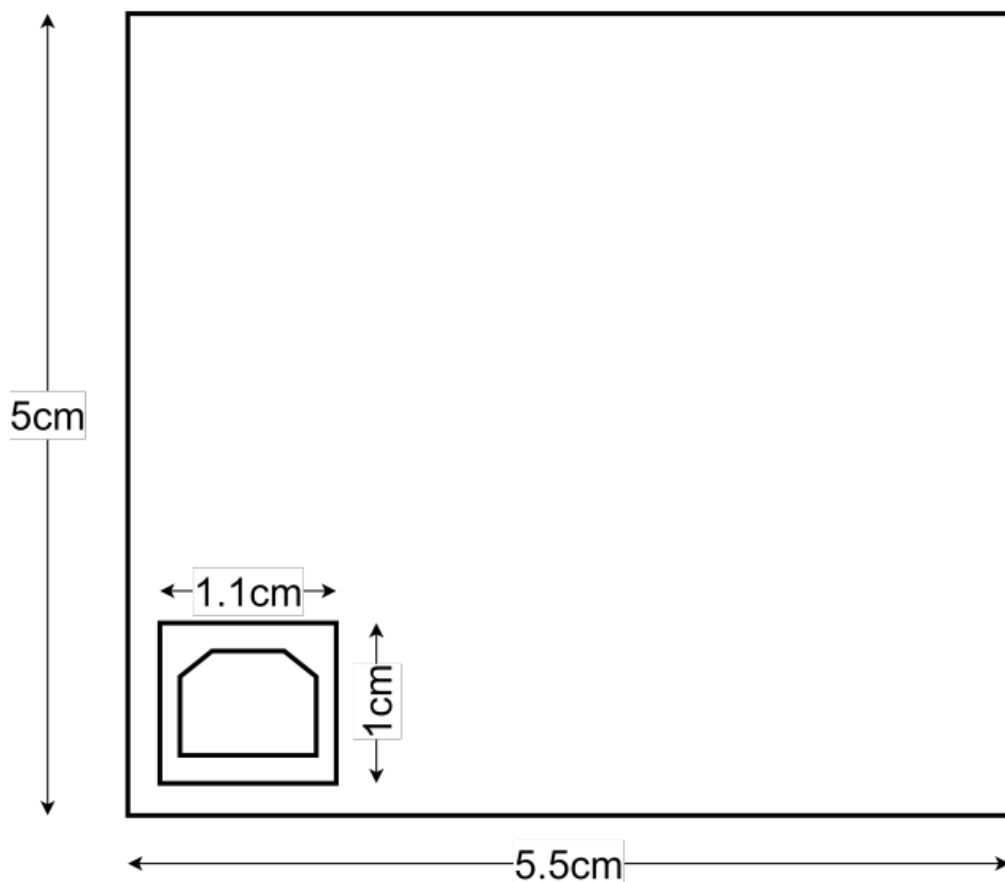
در ابتدا و در محاسبه هزینه بالا برای بسته بندی محصول یک کیس آردوینو در نظر گرفته شده بود که می توان آن را از [لینک](#) خریداری کرد. از مزایای استفاده از این بسته بندی جلوگیری از تماس دست با قطعات الکترونیکی، افزایش امنیت، و افزایش طول عمر محصول می باشد.

بسته بندی نهایی محصول به صورت یک جعبه مستطیلی پلاستیکی با ابعاد $10\text{ cm} \times 5.5\text{ cm} \times 5\text{ cm}$ است که این ها به ترتیب طول، عرض و ارتفاع محصول هستند. در بالای این جعبه مستطیلی، باید مطابق شکل ۲۴ تعداد ۲۰ سوراخ دایره ای برای پرت های ورودی و خروجی این سیستم وجود داشته باشد که در این سوراخ ها جک های موزی قرار می گیرند. ۱۶ جک برای ورودی های سیستم، ۲ جک برای خروجی ها و دو جک دیگر برای خروجی های GND و VCC اند. درون جعبه، هر کدام از جک ها با سیم کشی به پین مربوطه در آردوینو متصل می شود. نحوه سیم کشی درون جعبه طبق شکل ۵ و شکل ۲۴ مشخص است. برای استفاده از این جک ها، در بسته بندی کلی محصول علاوه بر این جعبه و سیم اتصال به رایانه (USB) باید سیم های پراب نیز قرار بگیرد که پراب مناسب این نوع جک برای خرید موجود است.



شکل ۲۴ نمای کلی جعبه از بالا

مطابق شکل ۲۵ در سمتی از جعبه که مجاور پورت سریال آردوینو است، سوراخی باید ایجاد شود که بتوان از طریق آن به پورت سریال برای اتصال قطعه آردوینو به رایانه دسترسی داشت. ابعاد این سمت جعبه و سوراخ در شکل مشخص شده است. به جز این دو سمت، بقیه سمت‌های جعبه بدون تغییر خاصی قسمتی مستطیلی با ابعاد مشخص شده اند.



شکل ۲۵ نمای کناری جعبه

برای بسته‌بندی نهایی محصول، جعبه‌ای مقوایی شامل جعبه بالا و سیم‌های مورد استفاده از سیستم مناسب است که ابعاد آن باید کمی بزرگ‌تر از ابعاد جعبه اصلی سیستم باشد.