آزمایشگاه سخت‌افزار

فانکشن ژنراتور و اسیلوسکوپ نرم افزاری

گزارش دوم

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم سال دوم ۰۱ ـ ۰۰

استاد:

جناب آقای دکتر اجلالی

اعضای تیم :

محمدمهدی‌جراحی ـ ۹۷۱۰۵۸۴۴

نگین‌جعفری ـ ۹۷۱۰۵۸۵۵

مهساامانی ـ ۹۷۱۰۵۷۶۹

فهرست مطالب

فهرست تصاویر

فهرست برنامه‌ها

۱ مراحل انجام بخش اول پروژه

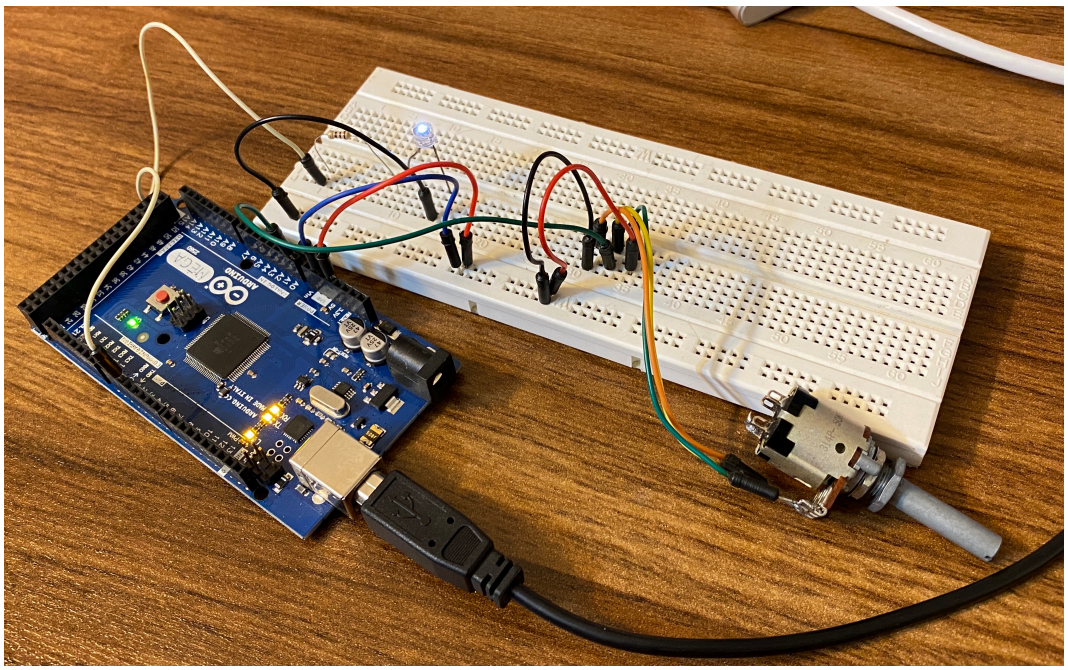
پروژه ما از دو بخش تشکیل می‌شود: بخش نمونه‌برداری (سخت‌افزاری) و بخش رسم نمودار (نرم‌افزاری).
در زیر به توضیح کارهای انجام‌شده‌ی سخت‌افزار و نرم‌افزار به صورت جداگانه می‌پردازیم.

۱.۱ سخت‌افزار

سخت‌افزار ما از یک برد آردوئینو به عنوان میکروکنترلر، مداری در کنار آردوئینو، و کد ریخته‌شده روی آردوئینو تشکیل شده‌است.

۱.۱.۱ مدار سخت‌افزار

به مدار فاز قبل، یک عدد LED به پایه‌ی PWM اضافه کردیم.



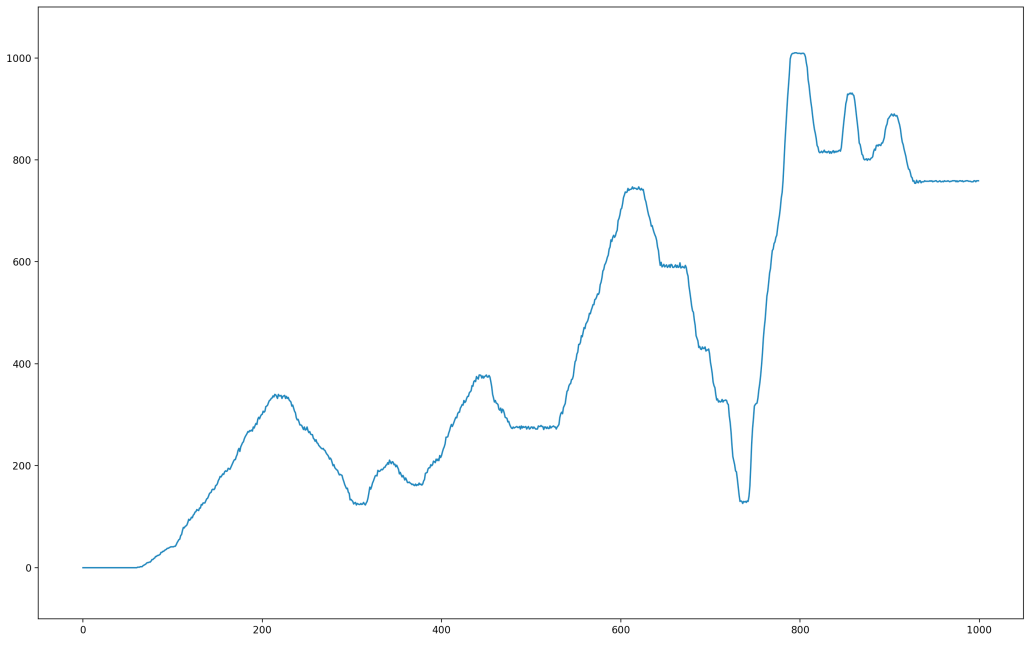
شکل ۱: مدار تکمیل‌شده شامل ولوم و LED

۱.۱.۲ کد سخت‌افزار

در گام بعد، با تغییر ساختار کد سخت‌افزار و نرم‌افزار و تغییر توابع نمونه‌گیری، تعداد نمونه در هر ثانیه را به شدت افزایش دادیم تا به ۱۰۰ نمونه در ثانیه برسد. دلیل اینکه تعداد نمونه را بیش از این افزایش ندادیم، محدودیت‌های پردازشی و حافظه برای ذخیره‌سازی آن‌ها بوده‌است. همچنین امکان دریافت عدد از پورت ارسال و قرار دادنش روی پورت PWM را اضافه کردیم.

کد به شرح زیر است.

```
int ledPin = 2;
String str;
int x;

void setup() {
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  delay(10);

  int val = analogRead(0);
  Serial.println(val);


  if (Serial.available() > 0) {
    str = Serial.readStringUntil('\n');
    x = Serial.parseInt();
    analogWrite(ledPin, x / 4);
    // analogRead values go from 0 to 1023, analogWrite values
    from 0 to 255
  }
}
```



شکل ۲: نمودار رسم شده از داده‌های آنالوگ دریافتی

در گام بعد برای تست سخت‌افزار، نرم‌افزار را به گونه‌ای تغییر دادیم که همان نمونه‌ی دریافتی آنالوگ را به سمت برد ارسال کند تا در خروجی تولید کند. فیلم این بخش در اینجا قابل مشاهده است.

با استفاده از قطعه کد زیر داده‌های چاپ شده روی پورت سریال را استخراج می‌کنیم. همچنین داده‌های مورد نیاز را روی پورت سریال ارسال می‌کنیم تا سخت‌افزار آن‌ها را تولید کند.

```
import queue
import threading
import serial
import random
import time

SAMPLE_DELAY = 10  # milli
buffer_lock = threading.Lock()
adc_inputs = []
pwm_queue = queue.Queue()


####  REAL  ####
def read_adc(ser: serial.SerialBase):
    try:
        sample = ser.readline().strip()
        if sample.decode("utf-8").isnumeric():
            sample = int(sample)
            with buffer_lock:
                adc_inputs.append(sample)
                pwm_queue.put(sample)  # Just for test
    except Exception as e:
        print(e)


def write_adc(ser: serial.SerialBase):
    with buffer_lock:
        if pwm_queue.empty():
            val = 0
        else:
            val = pwm_queue.get()
    ser.write(str.encode('{}\n'.format(val)))


def start_read_adc_thread():
    def continuous_read_adc():
        ser = serial.Serial(port='/dev/cu.usbmodem14201',
   baudrate=115200, timeout=1)
        while True:
            read_adc(ser)
```

```
            write_adc(ser)

    threading.Thread(target=continuous_read_adc).start()

####  FAKE   ####
# x = 500
# s = 0
#
#
# def read_adc():
#     global x, s
#     time.sleep(SAMPLE_DELAY / 1000)
#     s += random.randint(-3, 3)
#     s = min(s, 6)
#     s = max(s, -6)
#     x += s
#     x = min(x, 1024)
#     x = max(x, 0)
#     with buffer_lock:
#         adc_inputs.append(x)
#         pwm_queue.put(x)  # Just for test
#
#
# def write_adc():
#     with buffer_lock:
#         if pwm_queue.empty():
#             val = 500
#         else:
#             val = pwm_queue.get()
#     print('Output: ', val)
#
#
# def start_read_adc_thread():
#     def continuous_read_adc():
#         while True:
#             read_adc()
#             write_adc()
#
#     threading.Thread(target=continuous_read_adc).start()
```

۲.۱ نرم‌افزار

در بخش نرم افزاری برای اینکه بتوانیم داده ها را ذخیره و مجددا مشاهده کنیم، تابع live_plotter را مطابق زیر تغییر می‌دهیم :

```
# plot live data
def live_plotter():
    ani = FuncAnimation(plt.gcf(), animate, interval=INTERVAL)
    plt.tight_layout()

    save_button.on_clicked(save)
    transfer_button.on_clicked(transfer_to_arduino)

    plt.show()
```

همانند بخش قبلی برای نمایش نمودار زنده از ماژول FuncAnimation استفاده می کنیم. علاوه بر نمایش نمودار، نیاز داریم تا زمانی که می‌خواهیم داده ها را ذخیره کنیم و سپس در صورت نیاز آن ها را مشاهده کنیم. بدین منظور از ماژول CheckButtons کتابخانه matplotlib استفاده می کنیم. با اضافه کردن این دو دکمه به صفحه و بررسی وضعیت آن ها در تابع بالا(زمانی که کاربر روی آن ها کلیک می کند.)، کاربر می تواند تسک دلخواهش را انجام دهد.
سایز و محل قرار گیری این دکمه ها توسط قطعه کد زیر در ابتدای کد تعریف شده است:

```
# x position, y position, width and height
save_button = CheckButtons(plt.axes([0.5, 0.001, 0.5, 0.5],
   frame_on=False), ["save"], [False])
transfer_button = CheckButtons(plt.axes([0.5, 0.5, 0.5, 0.5],
   frame_on=False), ["observe saved data"], [False])
```

حال کاربر هنگامی که بر روی دکمه save کلیک کند، تابع زیر اجرا خواهد شد:

```
def save(label):
    global save_start_index
    if save_button.get_status()[0]:
        save_start_index = len(adc_inputs)
    else:
        vals = []
        f = open('values.txt', 'r')
        if os.stat("values.txt").st_size != 0:
            vals.extend(json.loads(f.read()))
        f.close()
        vals.extend(adc_inputs[save_start_index:])
        f = open('values.txt', 'w')
        f.write(json.dumps(vals))
        f.close()
```

با کلیک بر روی این دکمه برای شروع ذخیره سازی داده ها، بخش اول این تابع یعنی if اجرا می شود و تعداد ورودی های تا آن لحظه را ذخیره می کند تا بعدا مقادیر جدیدتر را با شروع از آن ایندکس بخواند. هنگامی که دیگر نخواهیم

داده ها را ذخیره کنیم، با فشردن دوباره این دکمه بخش else اجرا می شود و تمامی مقدار جدید ذخیره شده را در فایل values.txt ذخیره می شود. البته قابل ذکر است که در صورتی که از قبل مقادیری در این فایل ذخیره شده باشند، مقادیر جدید به انتهای آن ها اضافه می شوند.
در مرحله بعد برای اینکه کاربر بتواند داده های ذخیره شده اش را مشاهده کند باید بر روی دکمه observe saved data کلیک کند که در آن صورت تابع زیر اجرا می شود:

```
def transfer_to_arduino(label):
    if transfer_button.get_status()[0]:
        vals = []
        f = open('values.txt', 'r')
        if os.stat("values.txt").st_size != 0:
            vals.extend(json.loads(f.read()))
        f.close()
        with buffer_lock:
            pwm_queue.queue.clear()
            [pwm_queue.put(i) for i in vals]
    else:
        with buffer_lock:
            pwm_queue.queue.clear()
```

با کلیک بر روی این دکمه جهت مشاهده مقادیر ذخیره شده در فایل values.txt، این مقادیر از این فایل خوانده می شوند و و در صف pwm_queue قرار می گیرند. پس از اتمام مشاهده دوباره با کلیک بر روی آن تمام مقادیر قبلی از صف پاک می شوند تا بعدا در صورت فشردن این دکمه مقادیر جدید ذخیره شده را بر روی LED مشاهده کنیم.

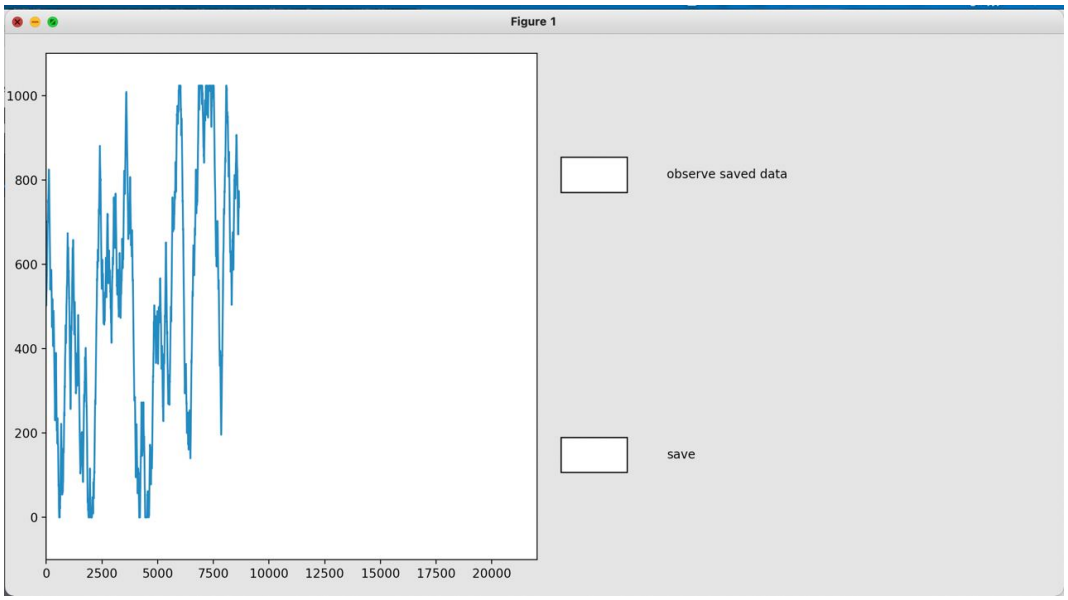
در نهایت فایل plotter.py بصورت زیر خواهد بود:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from arduino_serial import *
import numpy as np
from matplotlib.widgets import CheckButtons
import json, os

# for hiding buttons
mpl.rcParams["toolbar"] = "None"
INTERVAL = 100  # Time between graph frames in milli seconds.
optimal_frequency = (INTERVAL // SAMPLE_DELAY) * 2

fig = plt.figure(figsize=(12, 6), facecolor='#DEDEDE')
ax = plt.subplot(121)

# x position, y position, width and height
save_button = CheckButtons(plt.axes([0.5, 0.001, 0.5, 0.5],
   frame_on=False), ["save"], [False])
transfer_button = CheckButtons(plt.axes([0.5, 0.5, 0.5, 0.5],
   frame_on=False), ["observe saved data"], [False])
```

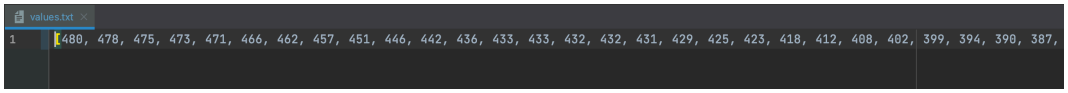
```
20
21 # clear content of values file
22 open('values.txt', 'w').close()
23
24 save_start_index = 0
25
26
27 def save(label):
28     global save_start_index
29     if save_button.get_status()[0]:
30         save_start_index = len(adc_inputs)
31     else:
32         vals = []
33         f = open('values.txt', 'r')
34         if os.stat("values.txt").st_size != 0:
35             vals.extend(json.loads(f.read()))
36         f.close()
37         vals.extend(adc_inputs[save_start_index:])
38         f = open('values.txt', 'w')
39         f.write(json.dumps(vals))
40         f.close()
41
42
43 def transfer_to_arduino(label):
44     if transfer_button.get_status()[0]:
45         vals = []
46         f = open('values.txt', 'r')
47         if os.stat("values.txt").st_size != 0:
48             vals.extend(json.loads(f.read()))
49         f.close()
50         with buffer_lock:
51             pwm_queue.queue.clear()
52             [pwm_queue.put(i) for i in vals]
53     else:
54         with buffer_lock:
55             pwm_queue.queue.clear()
56
57
58 # plot live data
59 def live_plotter():
60     ani = FuncAnimation(plt.gcf(), animate, interval=INTERVAL)
61     plt.tight_layout()
62
63     save_button.on_clicked(save)
64     transfer_button.on_clicked(transfer_to_arduino)
```

```
65
66      plt.show()
67
68
69  # animating each input data
70  def animate(i):
71      global ax
72      with buffer_lock:
73          inputs_copy = adc_inputs.copy()
74      if len(inputs_copy) == 0:
75          return
76      ax.cla()
77      ax.set_ylim(-100, 1100)
78      ax.set_xlim(0, np.power(np.e, int(np.log(len(inputs_copy))
    ) + 1))
79      ax.plot(inputs_copy)
80      # plt.plot(inputs_copy)
81      # # clear axis
82      # plt.cla()
83      # # plot data
84      # plt.scatter(len(inputs_copy) - 1, inputs_copy[-1])
85      # # show the data on the plot
86      # plt.text(len(inputs_copy) - 1, inputs_copy[-1] + 2,
    "{}".format(inputs_copy[-1]))
```

در سه تصویر زیر می توان نتایج اجرای این کد نرم افزاری را مشاهده کرد:



شکل ۳: نمودار رسم شده از داده‌های ورودی به همراه دکمه های اضافه شده



شکل ۴: مقادیر ذخیره شده در فایل