Spectrometer Interfaced with Arduino

OBJECTIVE

Write a Python script to control a spectrometer interfaced with an Arduino microcontroller. Wow!

INTRODUCTION (read this, then carefully follow the instructions in the subsequent sections)

The SPEX 1403 spectrometer separates a light beam according to wavelength. The basic ray diagram is shown in Figure 1.

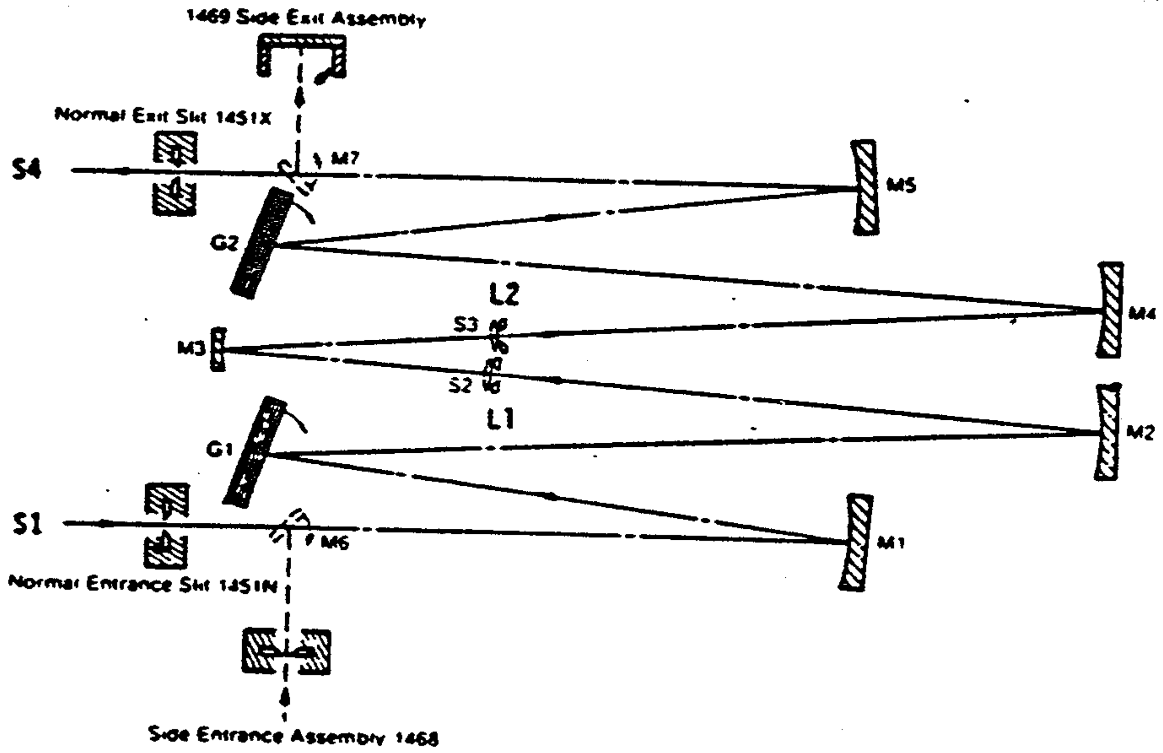
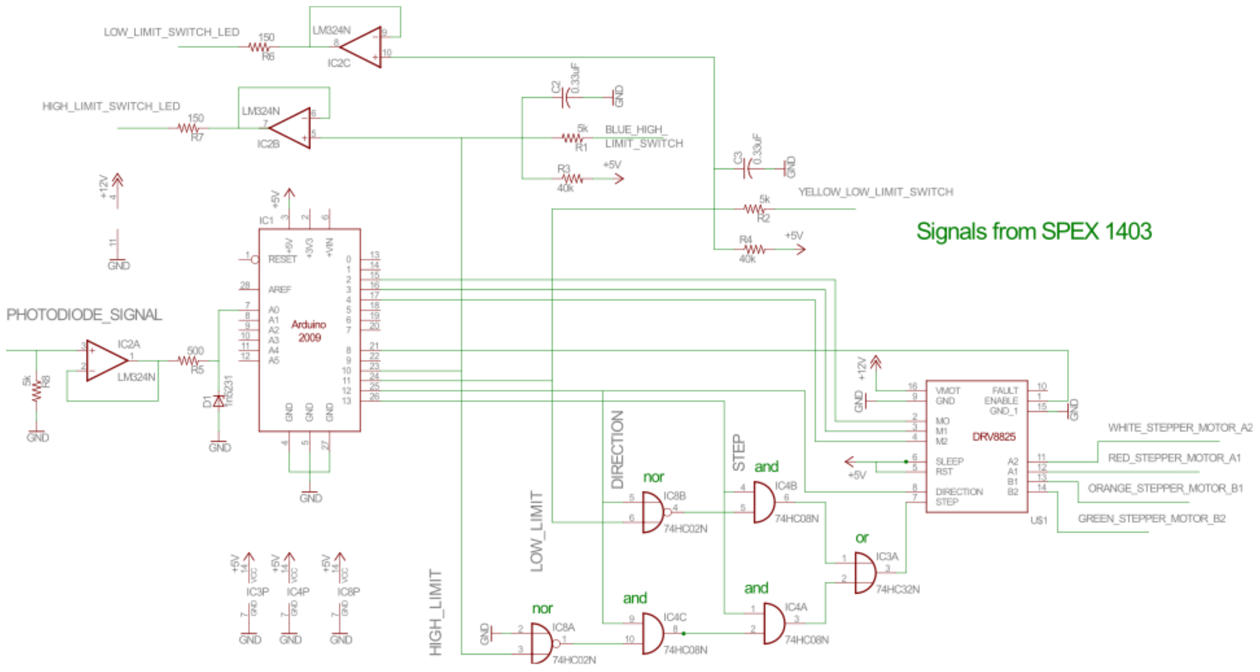


Figure 1. Optical diagram from the SPEX 1403/1404 manual.

The light enters the apparatus through slit S1 and then reflects off mirror M1. The light then reflects off diffraction grating G1, which separates the light by wavelength. The light then reflects off mirrors M2, M3, and M4. The light reflects off a second diffraction grating, G2, and then a final mirror, M5. A very narrow band of wavelengths passes through the slit S4 and into a photodiode.

What wavelength is detected by the photodiode? It depends on the angular position of the diffraction gratings. A stepper motor rotates both diffraction gratings (in sync with each other). As the diffraction gratings rotate, different wavelengths are directed through S4 into the photodiode. Thus, we can measure the spectrum of a light source by rotating the diffraction gratings.

The original electronics in this old instrument stopped working. Luckily, instrumentation genius Josh Savory (currently employed at the National Institute of Standards and Technology) rehabilitated the apparatus with new electronics. We will use the hardware he installed, but we will write our own software to control the apparatus. We need to send signals to the motor to make it move, and we need to input data from the photodiode. You'll develop skills in computer interfacing that can be applied to many apparatuses besides this one.



How do we connect the motor and the photodiode to a computer? Most computers don't have special ports for motors and photodiodes. However, modern computers all have USB ports. What we need is a device that can communicate through both a USB cable (connected to a computer) and simple wires (connected to the motor and photodiode). The Arduino microprocessor is a low-cost ($25) device that does just this. We will use Python to communicate with the Arduino, but there are many other programs that could perform the same task. (We used to use MATLAB.)

Figure 2. The circuit that interfaces with the photodiode, the motor, and the limit switches.

Dr. Savory's circuit is shown in Figure 2. Before we get too intimidated by this circuit diagram, let's state again that all we really need to do is: (1) receive data from the photodiode, and (2) tell the motor to move. Let's consider the photodiode first. The wire at the far left of the diagram (PHOTODIODE_SIGNAL) is the actual output of the photodiode. This wire is connected to an op amp circuit called a voltage follower. (It's good to understand the purpose of the voltage follower, but let's skip it for now.) Ultimately, the voltage from the photodiode is fed to the Arduino (analog input A0). The Arduino converts the analog voltage to a signal the computer can read.

Now look at the far right of Figure 2. We see four wires that connect to the stepper motor. Notice that they don't come directly from the Arduino. They come from an integrated circuit (a "chip") called a stepper motor controller. Four digital outputs from the Arduino are wired directly to the stepper motor controller. You'll need to understand how the stepper controller works, but for now, just recognize that the Arduino sends signals to the stepper controller, which sends voltages to the stepper motor.

Finally, we need to understand the limit switches. There's a limit to how far the diffraction gratings can rotate in either direction. To avoid damaging the apparatus, we have to make sure that the motor doesn't try to rotate the gratings past their limits. A digital "high" voltage (5 V, represented as "1") indicates that the motor has reached a limit. A "1" on the high limit switch indicates that we should not instruct the motor to turn in the forward direction. A "1" on the low limit switch indicates that we should not instruct the motor to turn in the backward direction. Actually, Dr. Savory's circuit automatically prevents the motor from turning past its limits. (That's the purpose of the logic gates, which look like semicircles in the circuit diagram.) However, to be safe, our Python program should use the limit-switch signals (fed into the Arduino) to make sure we don't try to rotate the motor past its limits.

So, we need to understand nine wires connected to the Arduino. There are six digital outputs that connect to the stepper controller (which controls the motor):

Pins 2, 3, 4 (M0, M1, M2): microstep controls; set to 0 to take full steps

Pin 8 ($\overline{\mathrm{EN}}$): set to 0 to enable motion

Pin 12 (DIR): 1 is forward, 0 is backward

Pin 13 (STEP): rising edge (transition from 0 to 1) causes motor to take one step

There's one analog input, which is the data from the photodiode:

A0: photodiode input (0-5V)

And there are two digital inputs from the limit switches:

Pin 10: high limit switch

Pin 11: low limit switch

Our Python script needs to record the voltages at the Arduino inputs and send digital voltages (0 or 1) to the Arduino outputs. No problem!

ARDUINO

A program (called a sketch) runs on the Arduino. The sketch will execute even if the Arduino is not connected to a computer. We could learn how to write sketches. Instead, let's just use the sketch that a Python programmer wrote to enable Python to communicate easily with an Arduino. The sketch is called "prototype.pde." It may already be on the two Arduino boards that we'll be using. (One is in the electronics box and wired to the hardware as shown in Figure 2, and the other you'll be practicing on.) If you need to upload prototype.pde to an Arduino, open this file in the Arduino software and select File → Upload to I/O. Always close the Arduino software before running a Python script that communicates with the Arduino.

Unplug the USB cable that connects the computer to the Arduino in the electronics box. Plug in the practice Arduino instead. Use the practice Arduino to learn the Python functions in the following section.

PYTHON

We need to learn only seven special Python functions to communicate with the Arduino! These functions are defined in the arduino.py file that's already on the computer (downloaded from https://github.com/vascop/Python-Arduino-Proto-API-v2/archive/master.zip). To give yourself access to these functions, open Spyder and in the Python console type

```
from arduino.arduino import Arduino
```

To establish communication with the Arduino:

```
b=Arduino('COM4')
```

"b" is an arbitrary name for the Arduino. The computer assigns a "COMmunications port" number to the Arduino (not always 4). To determine the port number, run the Device Manager.

The next communication with the Arduino MUST be the specification of which digital pins are outputs, for example:

```
b.output([2,3])  #if you want digital pins 2 and 3 to be outputs.  The rest are inputs by default.
```

Here are the input and output functions:

Digital input:

```
x=b.getState(8) #This sets x equal to the digital input (False for 0 V, True for 5 V) on pin 8.
```

Digital output:

```
b.setLow(2)     #This outputs 0 V on pin 2.

b.setHigh(3)    #This outputs 5 V on pin 3.
```

Analog input:

```
y=b.analogRead(0)       #This sets y equal to a number proportional to the voltage on pin A0.
```

Terminate the connection to the Arduino:

```
b.close()
```

Write Python scripts to practice using all of these functions with the practice Arduino board. Use voltmeters and voltage sources to make sure everything is working as you expect. Do you know how to plot a changing analog input? Do you know how to average five measurements of analog input, record the average, and then repeat 100 times?

This will give you access to useful functions:

```
import numpy as np, matplotlib.pyplot as plt
```

Here's a partial list of Python functions you'll need:

You can create an array of 10 zeros with

```
z=np.zeros(10)
```

You can create the array [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] with

```
zerothroughnine=np.arange(10)
```

You can replace the zeros in z with ten measured voltages:

```
for a in zerothroughnine :
    z[a]=b.analogRead(0)
```

You can plot the result with

```
plt.plot(zerothroughnine,z)
```

If you need any help with Python, just ask (me or google).

STEPPER CONTROLLER

Now wire the (practice) stepper controller to the (practice) Arduino and the (practice) stepper motor. You'll need to study the spec sheet for the stepper controller (http://www.pololu.com/product/2134). Write a Python script that controls the motor.

The spectrometer is wired to a higher-current stepper controller (http://www.pololu.com/product/2133), which is otherwise similar to the (practice) stepper controller.

The wiring diagram for the practice stepper motor is here: http://www.orientalmotor.com/products/pdfs/opmanuals/HM-601-13JECK.pdf. (English on p. 2.)

SPECTROMETER

Using what you've learned about Arduino, Python, and stepper controllers, write a Python script to rotate the diffraction gratings and record data from the photodiode! We need a bright light source, so you'll probably want to use the Na lamp or the HeNe laser. Ultimately, you should calibrate the spectrometer: from a given starting point, what wavelength corresponds with the number of steps taken by the motor? You can use the "wavenumber" display to record a starting point, but the displayed "wavenumber" is not an accurate indication of inverse wavelength.

You may assume that the wavelength λ is linearly related to the inverse of nominal "wavenumber" N:

$$\lambda = m(1/N) + b$$

Your job is to determine m and b. Two known wavelengths are required to do this. You can use the HeNe laser and one of the lines of the Na doublet. Then, test your calibration with the other line of the Na doublet.

To understand how the rotation of the motor determines the wavelength at the exit slit, we need to study Fig. 3 (on the next page), an excerpt from the manufacturer's manual. I believe the stepper motor rotates the leadscrew, LS. Why is the apparatus in Fig. 3 called a cosecant drive? The rotation of the motor (the number of steps taken) is proportional to what function of the rotation of the diffraction gratings? The rotation of the motor is proportional to what function of the wavelength at the exit slit? You need to answer this question to calibrate the spectrometer!

As you study Fig. 3, it may be helpful to recognize that α and β appear to be interchanged: β appears to be the angle of incidence, and α appears to be the angle of diffraction.

Do your best to explain Fig. 3 in your lab report. Also, can you explain Fig. 2, specifically, the purpose of the op amps and the logic gates? Can you briefly (or thoroughly) explain the physics of a stepper motor?

4.1 Fundamental grating equation applied to Czerny-Turner mount

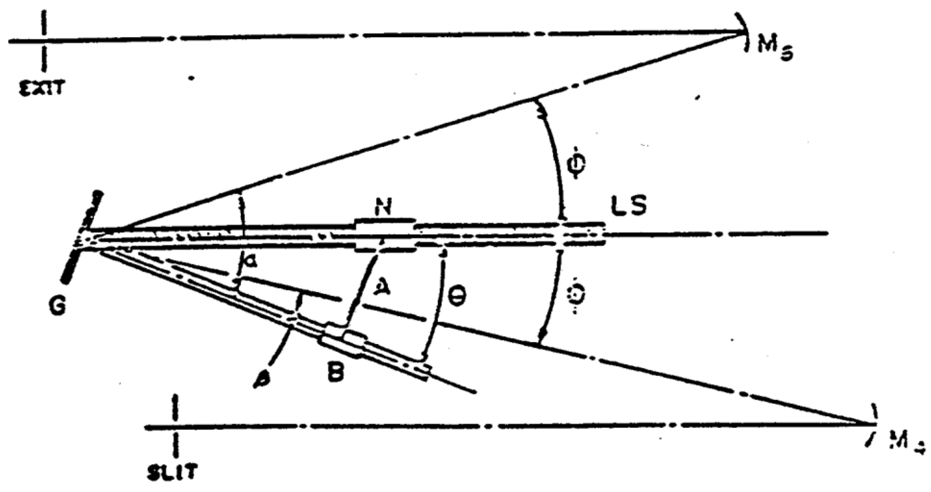
$m\lambda = d(\sin\alpha + \sin\beta)$ where

m = order
λ = wavelength
d = grating spacing
α = angle of incidence
β = angle of diffraction

For convenience in the 1403/1404 instruments, this may be expressed as

$m\lambda = (2d \sin\theta \cos\phi)$

$\phi = 10^{\circ}$ and $\cos\phi = 0.984$
θ = grating rotation measured from zero, its position at the direct image
$\alpha = \theta + \phi$
$\beta = \theta - \phi$



G Grating	B Slide
N Nut	A Arm
LS Leadscrew	M_4 and M_5 Mirrors

In the 1403 Cosecant Drive the nut N moves along leadscrew LS, while the slide B moves along a precision bar beld at right angles to the grating G. The arm A is held at right angles to the bar and the grating is thus rotated as the arm A moves along the bar. The 1404 has a similar Sine Drive.

Figure 3. Excerpt from the manufacturer's manual explaining the rotation of the diffraction gratings.