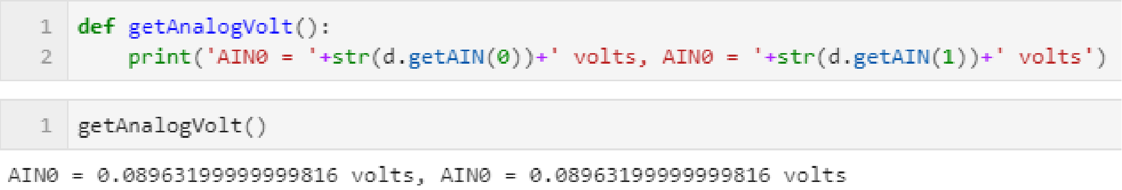
Solutions to Assignment 4 by Aether Zhou
This file is
https://docs.google.com/document/d/1HrPdf9-wTB6-ptsPKQa7UNZuBqBlAEtdnr3rNexXug4/edit?usp=sharing
Python filename is assignment_four_aether_zhou.py

1) Write a program that measures both analog in values and prints the results in a human readable way.

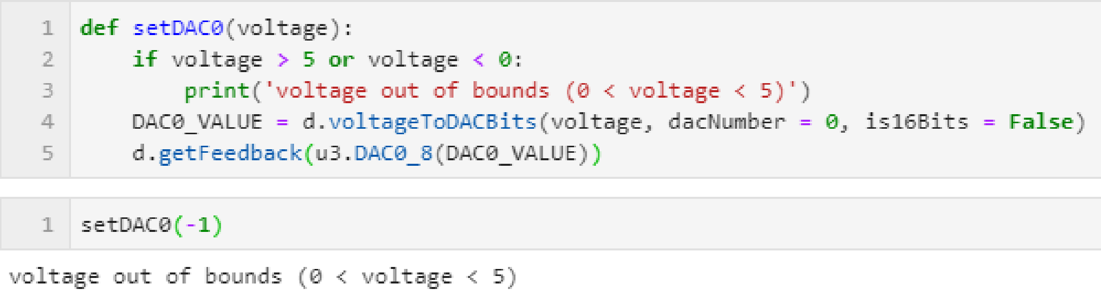


```
def getAnalogVolt():
    print('AIN0 = '+str(d.getAIN(0))+' volts, AIN0 = '+str(d.getAIN(1))+' volts')
```

```
getAnalogVolt()
```

```
AIN0 = 0.08963199999999816 volts, AIN0 = 0.08963199999999816 volts
```

2) Write a program setDAC0(voltage) that sets the DAC0 voltage, and responds gracefully if you send it an illegal voltage (say, greater than 5 volts, or less than 0).

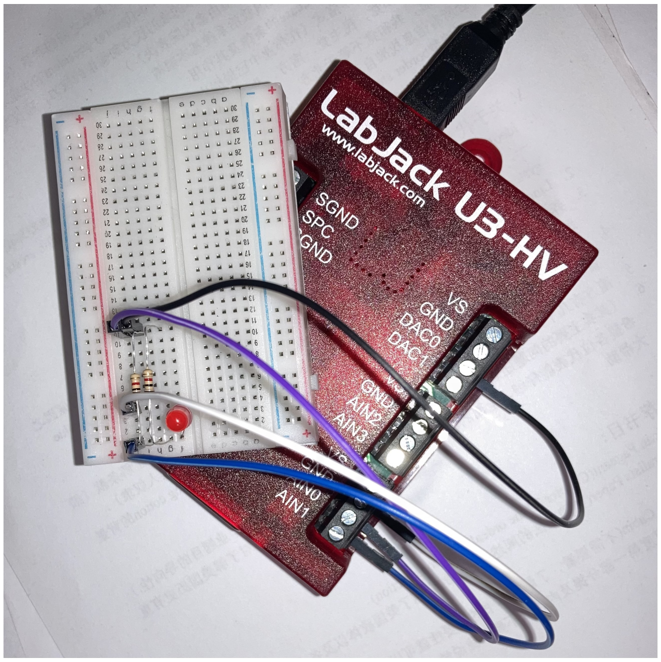


```
def setDAC0(voltage):
    if voltage > 5 or voltage < 0:
        print('voltage out of bounds (0 < voltage < 5)')
    DAC0_VALUE = d.voltageToDACBits(voltage, dacNumber = 0, is16Bits = False)
    d.getFeedback(u3.DAC0_8(DAC0_VALUE))
```

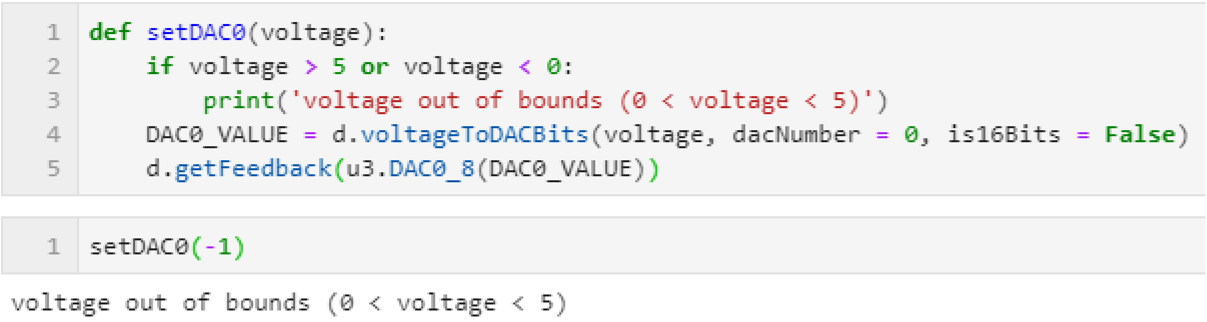
```
setDAC0(-1)
```

```
voltage out of bounds (0 < voltage < 5)
```

3) Make the following circuit, using your kits and the Labjack



4) use setDAC0(voltage) to turn the LED on and off manually.

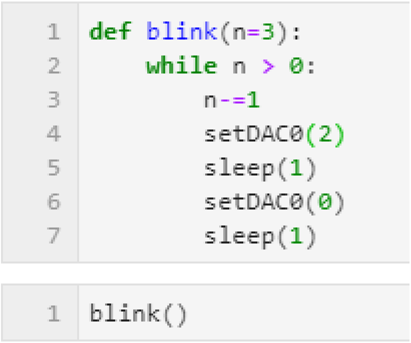


```
def setDAC0(voltage):
    if voltage > 5 or voltage < 0:
        print('voltage out of bounds (0 < voltage < 5)')
    DAC0_VALUE = d.voltageToDACBits(voltage, dacNumber = 0, is16Bits = False)
    d.getFeedback(u3.DAC0_8(DAC0_VALUE))
```

```
setDAC0(-1)
```

```
voltage out of bounds (0 < voltage < 5)
```

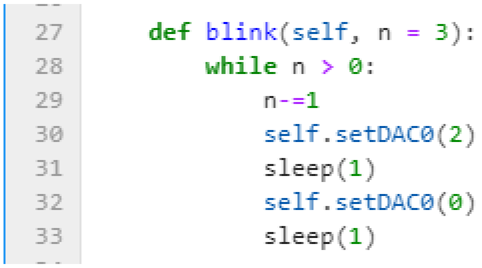
5) Write a program called blink(n) that blinks the LED n times.



```
def blink(n=3):
    while n > 0:
        n-=1
        setDAC0(2)
        sleep(1)
        setDAC0(0)
        sleep(1)
```

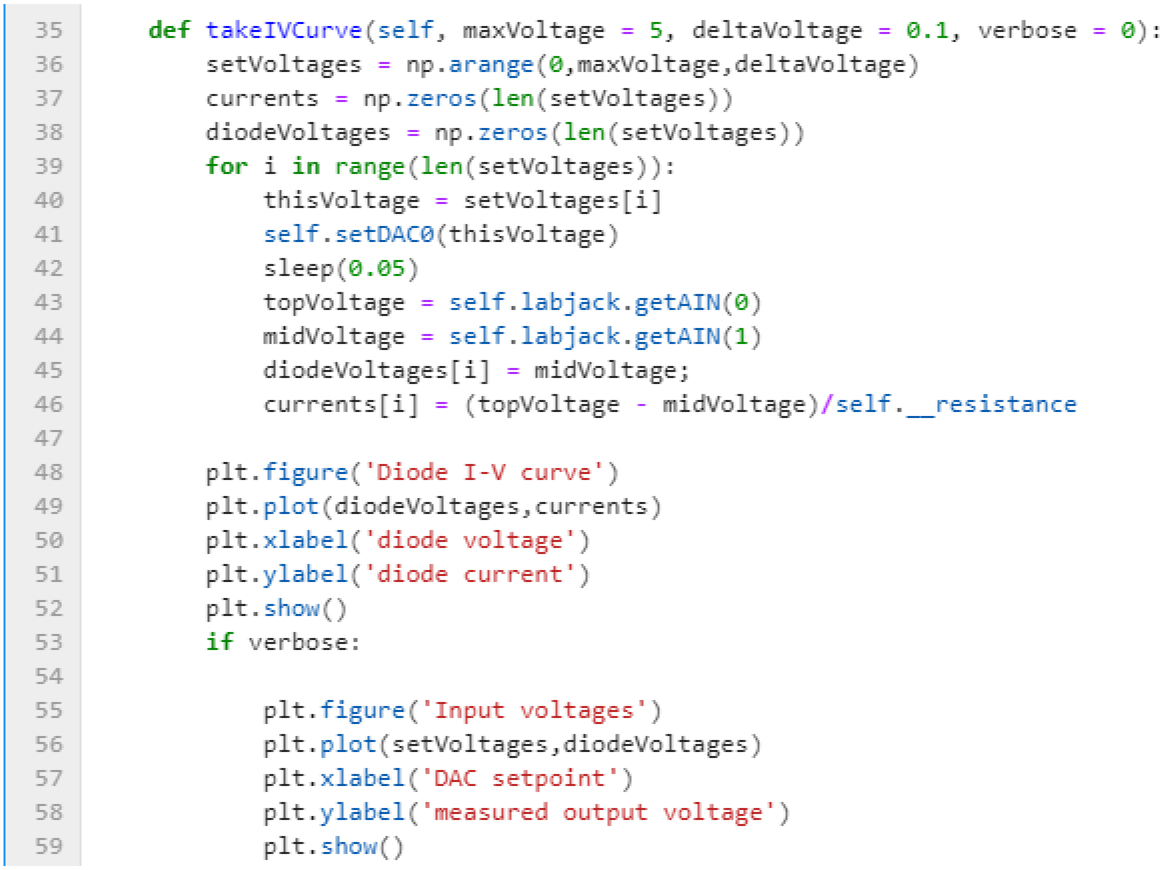
```
blink()
```

6) extend helloRobot.py to include blinker.

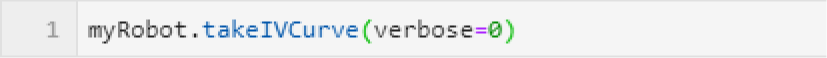


```
    def blink(self, n = 3):
        while n > 0:
            n-=1
            self.setDAC0(2)
            sleep(1)
            self.setDAC0(0)
            sleep(1)
```

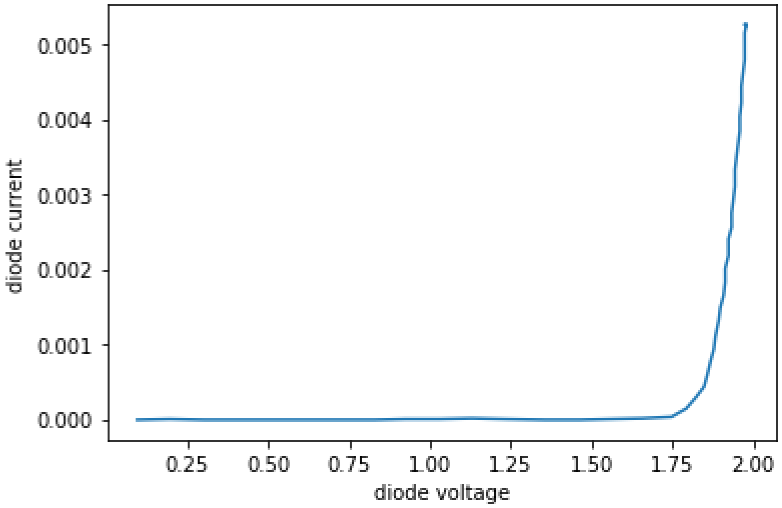
7) Now write takeIVCurve() which uses your circuit to measure the I-V curve of the diode.



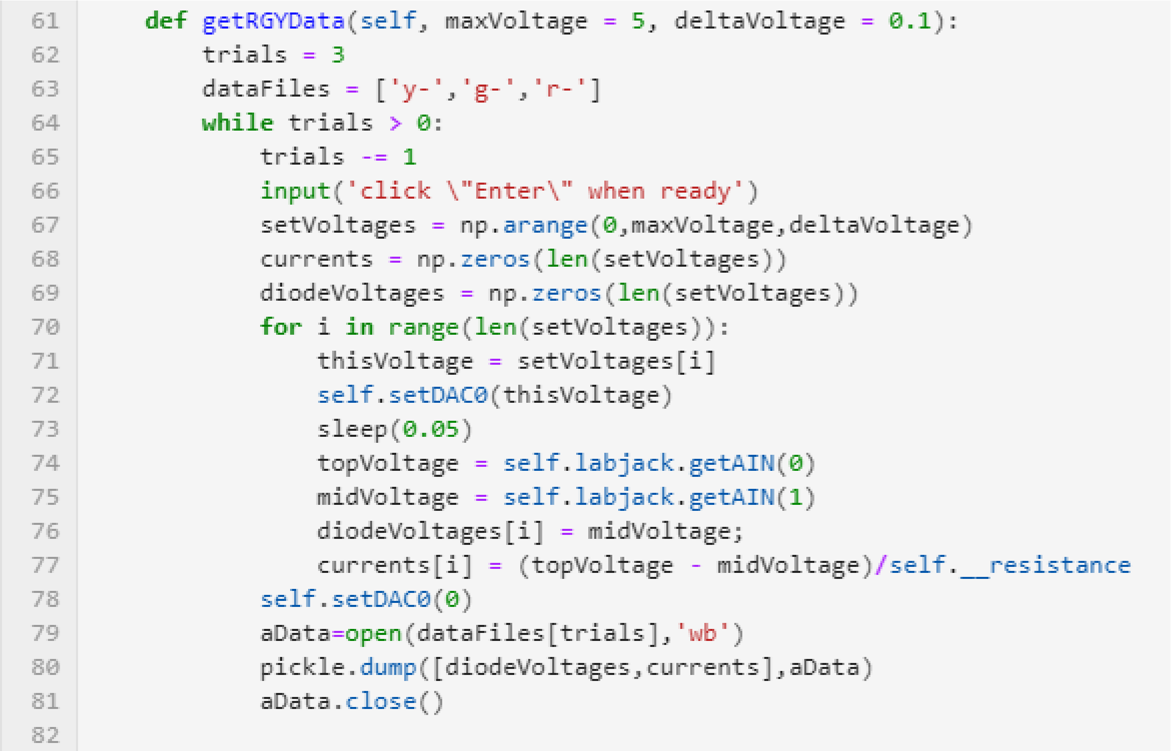
```
    def takeIVCurve(self, maxVoltage = 5, deltaVoltage = 0.1, verbose = 0):
        setVoltages = np.arange(0,maxVoltage,deltaVoltage)
        currents = np.zeros(len(setVoltages))
        diodeVoltages = np.zeros(len(setVoltages))
        for i in range(len(setVoltages)):
            thisVoltage = setVoltages[i]
            self.setDAC0(thisVoltage)
            sleep(0.05)
            topVoltage = self.labjack.getAIN(0)
            midVoltage = self.labjack.getAIN(1)
            diodeVoltages[i] = midVoltage;
            currents[i] = (topVoltage - midVoltage)/self.__resistance

        plt.figure('Diode I-V curve')
        plt.plot(diodeVoltages,currents)
        plt.xlabel('diode voltage')
        plt.ylabel('diode current')
        plt.show()
        if verbose:

            plt.figure('Input voltages')
            plt.plot(setVoltages,diodeVoltages)
            plt.xlabel('DAC setpoint')
            plt.ylabel('measured output voltage')
            plt.show()
```



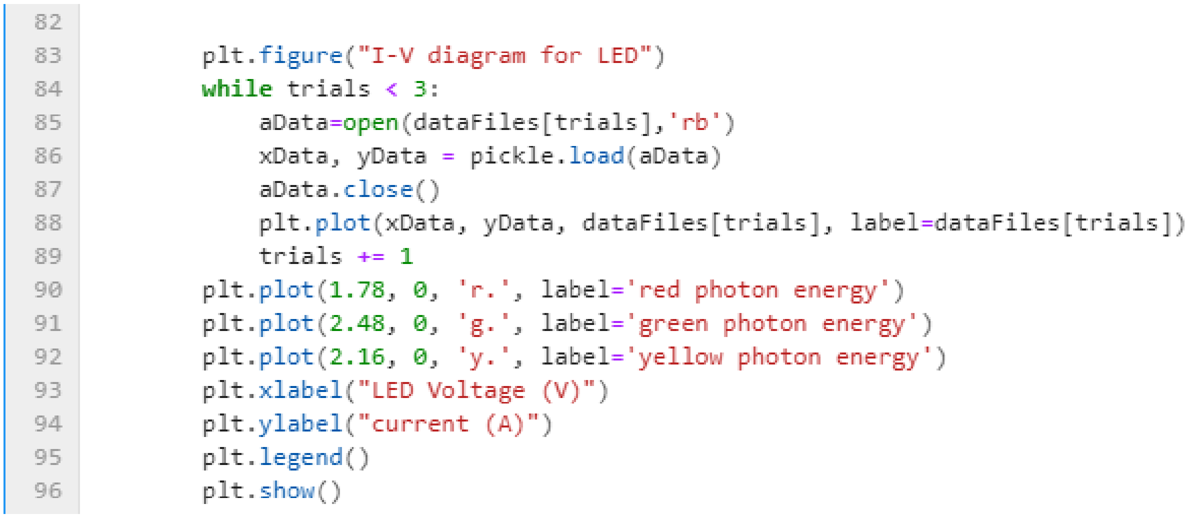
```
myRobot.takeIVCurve(verbose=0)
```



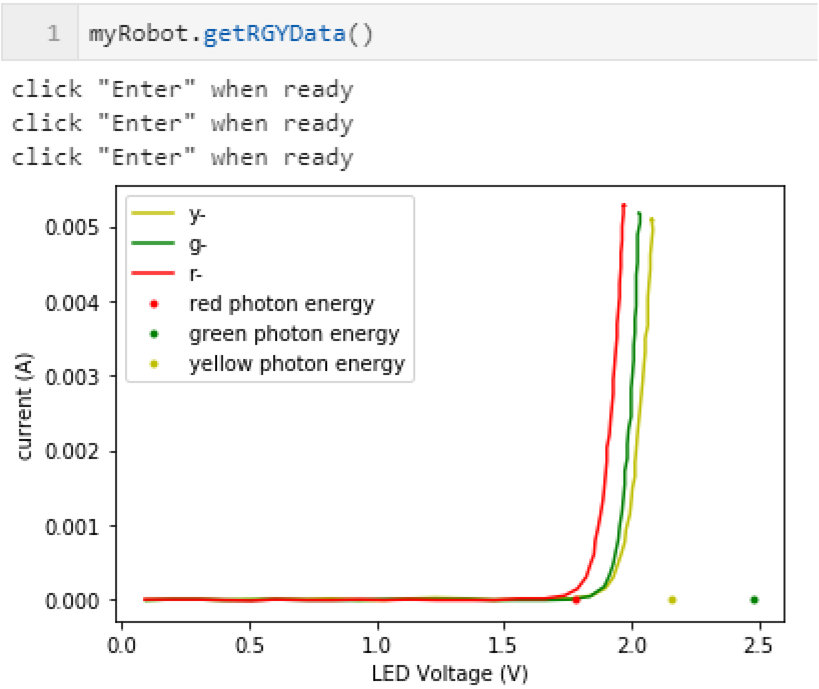
8) Now use the tools you have so far to make a plot like our initial example (you have less colors, but use what you've got). Make such a plot, with appropriate colors and labels.



```
    def getRGYData(self, maxVoltage = 5, deltaVoltage = 0.1):
        trials = 3
        dataFiles = ['y-','g-','r-']
        while trials > 0:
            trials -= 1
            input('click \"Enter\" when ready')
            setVoltages = np.arange(0,maxVoltage,deltaVoltage)
            currents = np.zeros(len(setVoltages))
            diodeVoltages = np.zeros(len(setVoltages))
            for i in range(len(setVoltages)):
                thisVoltage = setVoltages[i]
                self.setDAC0(thisVoltage)
                sleep(0.05)
                topVoltage = self.labjack.getAIN(0)
                midVoltage = self.labjack.getAIN(1)
                diodeVoltages[i] = midVoltage;
                currents[i] = (topVoltage - midVoltage)/self.__resistance
            self.setDAC0(0)
            aData=open(dataFiles[trials],'wb')
            pickle.dump([diodeVoltages,currents],aData)
            aData.close()

```



```

        plt.figure("I-V diagram for LED")
        while trials < 3:
            aData=open(dataFiles[trials],'rb')
            xData, yData = pickle.load(aData)
            aData.close()
            plt.plot(xData, yData, dataFiles[trials], label=dataFiles[trials])
            trials += 1
        plt.plot(1.78, 0, 'r.', label='red photon energy')
        plt.plot(2.48, 0, 'g.', label='green photon energy')
        plt.plot(2.16, 0, 'y.', label='yellow photon energy')
        plt.xlabel("LED Voltage (V)")
        plt.ylabel("current (A)")
        plt.legend()
        plt.show()
```



```
myRobot.getRGYData()
```

```
click "Enter" when ready
click "Enter" when ready
click "Enter" when ready
```

9) Using Google and whatever else you like, find the energy of a single red photon, green photon, and yellow photon.

E=hf, where h is the Planck's constant, f is the frequency of photon, and E is the energy. We can plug in the threshold voltage as E to calculate the h, although this calculation is only good for the red LED used in this experiment.

Full Code:

```
import u3
from time import sleep
import matplotlib.pyplot as plt
import numpy as np
import pickle

class robot:

    def __init__(self):
        self.labjack = u3.U3()
        self.__resistance = 500

    def close(self):
        self.labjack.close()

    def printAnalogIns(self):
        AIN0value = self.labjack.getAIN(0)
        AIN1value = self.labjack.getAIN(1)
        descriptor ='A0 = %5.3f volts, A1 = %5.3f volts ' %
(AIN0value, AIN1value)
        print(descriptor)

    def setDAC0(self,voltage = 0):
        if voltage < 0:
            print('output voltage must be >= 0')
            return
        elif voltage > 5:
            print('output voltage must be <= 5')
            return
        else:
            DAC0_VALUE = self.labjack.voltageToDACBits(voltage,
dacNumber = 0, is16Bits = False)
            self.labjack.getFeedback(u3.DAC0_8(DAC0_VALUE))

    def blink(self, n = 3):
        while n > 0:
            n-=1
            self.setDAC0(2)
            sleep(1)
            self.setDAC0(0)
            sleep(1)
    def takeIVCurve(self, maxVoltage = 5, deltaVoltage = 0.1, verbose
= 0):
        setVoltages = np.arange(0,maxVoltage,deltaVoltage)
        currents = np.zeros(len(setVoltages))
        diodeVoltages = np.zeros(len(setVoltages))
        for i in range(len(setVoltages)):
            thisVoltage = setVoltages[i]
            self.setDAC0(thisVoltage)
            sleep(0.05)
            topVoltage = self.labjack.getAIN(0)
            midVoltage = self.labjack.getAIN(1)
            diodeVoltages[i] = midVoltage;
            currents[i] = (topVoltage - midVoltage)/self.__resistance

        plt.figure('Diode I-V curve')
        plt.plot(diodeVoltages,currents)
        plt.xlabel('diode voltage')
        plt.ylabel('diode current')
        plt.show()
        if verbose:

            plt.figure('Input voltages')
            plt.plot(setVoltages,diodeVoltages)
            plt.xlabel('DAC setpoint')
            plt.ylabel('measured output voltage')
            plt.show()

    def getRGYData(self, maxVoltage = 5, deltaVoltage = 0.1):
        trials = 3
        dataFiles = ['y-','g-','r-']
        while trials > 0:
            trials -= 1
            input('click \"Enter\" when ready')
            setVoltages = np.arange(0,maxVoltage,deltaVoltage)
            currents = np.zeros(len(setVoltages))
            diodeVoltages = np.zeros(len(setVoltages))
            for i in range(len(setVoltages)):
                thisVoltage = setVoltages[i]
                self.setDAC0(thisVoltage)
                sleep(0.05)
                topVoltage = self.labjack.getAIN(0)
                midVoltage = self.labjack.getAIN(1)
                diodeVoltages[i] = midVoltage;
                currents[i] = (topVoltage -
midVoltage)/self.__resistance
```

```
            self.setDAC0(0)
            aData=open(dataFiles[trials],'wb')
            pickle.dump([diodeVoltages,currents],aData)
            aData.close()

        plt.figure("I-V diagram for LED")
        while trials < 3:
            aData=open(dataFiles[trials],'rb')
            xData, yData = pickle.load(aData)
            aData.close()
            plt.plot(xData, yData, dataFiles[trials],
label=dataFiles[trials])
            trials += 1
        plt.plot(1.78, 0, 'r.', label='red photon energy')
        plt.plot(2.48, 0, 'g.', label='green photon energy')
        plt.plot(2.16, 0, 'y.', label='yellow photon energy')
        plt.xlabel("LED Voltage (V)")
        plt.ylabel("current (A)")
        plt.legend()
        plt.show()

myRobot = robot()
myRobot.takeIVCurve(verbose=0)
myRobot.setDAC0(0)
myRobot.getRGYData()
myRobot.close()
```