Exam 1: Battery Discharge

Max Huggins – UCA Department of Physics Astronomy

February 18, 2019

Abstract

This experiment was done to determine the life of a 9V Energizer, Industrial battery. An analog to digital converter (ADC) chip was used to take voltage measurements across a resistor in a voltage dividing circuit. A voltage-time curve was plotted to show the relationship.

1 Introduction

When working with mobile electronics, some form of a battery is necessary for powering components. There are several important specifications for determining if a battery will work for its intended use. The specification of interest for this experiment is the discharge rate of the battery. This gives valuable information about how long the device will function and how long the battery can supply the necessary voltage for the job.
To determine the voltage as a function of time, the circuit in figure 3 was used. The ADC used was the ADC0831, which has a maximum reference voltage value of $5V$. This means a voltage dividing circuit was necessary to measure the voltage of a $9V$ battery. Resistors for the voltage divider were decided with an initial current of $45mA$ in mind and an output voltage (V_{out}) of no more than the reference voltage of the ADC ($5V$). Where half of a $9V$ battery is close to this value. Ohm's law was used to determine values for the resistors of the voltage divider.

$$R_1 + R_2 = \frac{V_{battery}}{I} \tag{1}$$

Typically, discharges are done at a continuous current to determine the life of the battery. In this experiment, the battery was discharged through two 100Ω resistors and the current varied due to the changing voltage of the battery. To determine the ampere hours of the battery a trapezoidal sum was used through each data point.

2 Analysis

The python script used for making figure 1 is shown at the end of this document and a flow chart for the code is available in the Experimental Setup section in figure 4.

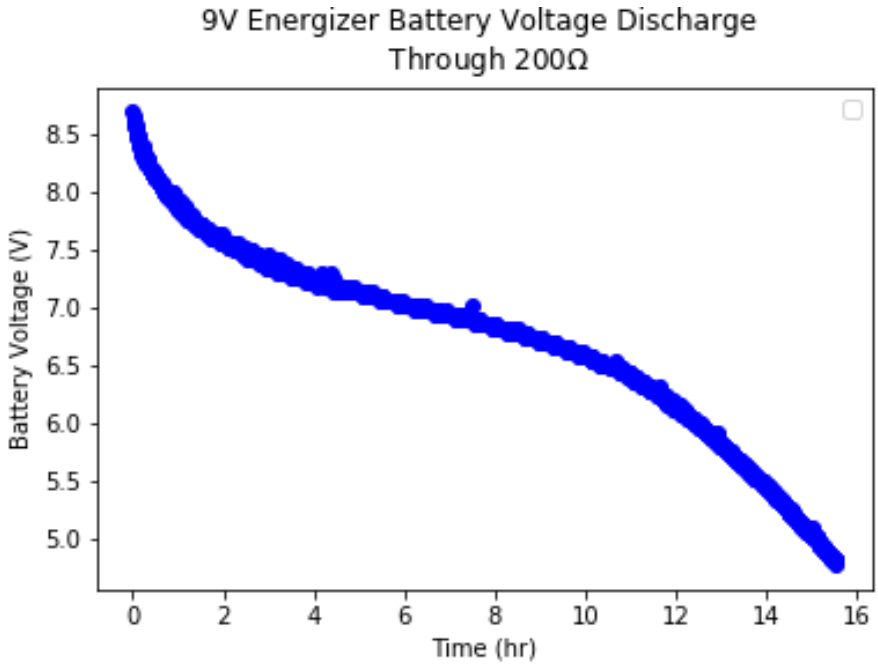

Figure 1: Here is the voltage versus time data plotted.

The trapezoidal sum used to determine the ampere hours of the battery works similarly to figure 2:

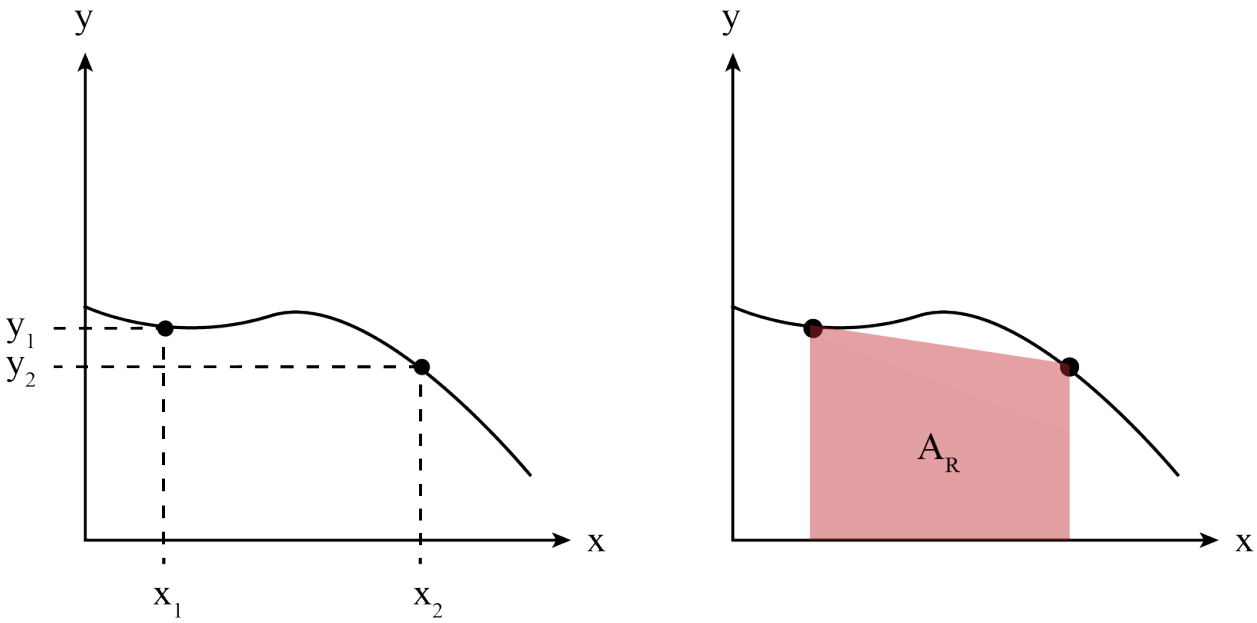

Figure 2: Here is the logic behind the trapezoidal sum.

Where the area of the red trapezoid is given by:

$$A_R = \frac{1}{2}(y_2 + y_1)(x_2 - x_1) \tag{2}$$

Except, the y axis is given by a current corresponding to the voltage data point taken at some time, plotted on the x-axis. A trapezoid is given for every two data points and the area is summed up to acquire the total amp-hours for the battery. The python script for this calculation is available at the end of the document. The capacity, or lifespan, was found to be $530mAh$. Since the battery's data sheet offers capacity data at only constant current discharges, the average current value of the data set was used as a reference. The average current value for this data set was $33mA$ and the corresponding continuous discharge drain from the data sheet for the 9V Energizer, Industrial battery shows a lifespan of approximately $500mA$. These are rough values and no further error analysis was done to describe the accuracy of this value, unfortunately.

3 Experimental Setup

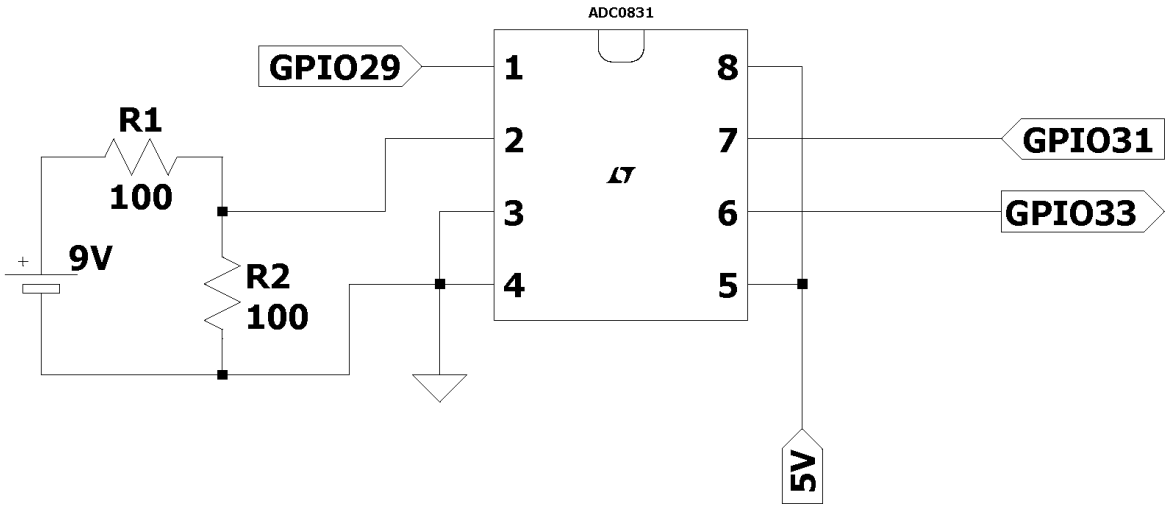

Figure 3: Here is the schematic for the voltage measuring circuit with the ADC0831

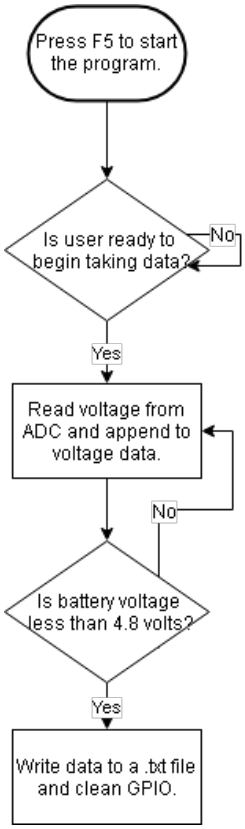

Figure 4: This is the flow chart for the code that records battery voltage and time data.

4 Conclusion

An analog to digital converter chip was used to take voltage data from a 9V Energizer Battery being discharged through two 100Ω resistors. From this, a voltage versus time plot was made and the capacity of the battery was measured.

Python script for data logging with the ADC.

```
#Exam 1
#Feb 11, 2019
#Max Huggins

import RPi.GPIO as GPIO
import time
import uControllersDataAcquisition as DtA

GPIO.setmode(GPIO.BOARD)
CS = 29
CLK = 31
DO = 33

#define a function to handle reading ADC data
d = DtA.readADC()

#define a function to return voltage
DtA.calc_voltsADC(d)

try:
    #Create data arrays
    time_data = []
    voltage_data = []
    #Begin test
    my_test = True
    ready = 'N'
    while ready == 'N':
        print("Are you ready to begin recording? (Y/N)")
        ready = input()
        if ready == 'Y':
            print("Make sure your battery is plugged in. Let's begin our
    journey!")
    start_time = time.time()
    time.sleep(1)
    while my_test == True:
        #append time data
        time_data.append(time.time() - start_time)
        #Call function to read a binary data value
        d = DtA.readADC() #This returns the binary data
        #append the binary data
        voltage_data.append(d)
        voltage = DtA.calc_volts(d)
        print(voltage)
        if voltage <= 2.4: #stop if under 2.4V (means 9V batt is at 4.8V)
            my_test = False
        else:
            time.sleep(30)
            print(voltage)
    #open a data file for writing in the same directory as the working program
    file = open('Test1NEWCode.txt', 'w')
    for n in range(len(time_data)):
        my_volts = DtA.calc_volts(voltage_data[n])
```

```
        #Write the data as comma delimites
        file.write(str(time_data[n]) + ',' + str(my_volts) + '\n')
    #always close the file you are using

    file.close()

except KeyboardInterrupt:
    print("it's fried, dude")

finally:
    GPIO.cleanup()
    print("Isaac cleaned the oven...")
```

Python script for plotting ADC data.

```
#Plotting Test data
#max huggins
#2/17/19

import matplotlib.pyplot as plt
import numpy as np

R_1 = 100
R_2 = 100

#define data arrays
time_data = []
voltage_data = []

#read in data
lines = np.loadtxt('Test_1.txt', delimiter=',')
for line in lines:
    time_data.append(line[0] / 3600)
    v_in = ((R_1 + R_2) / (R_1)) * line[1]
    voltage_data.append(v_in)

#make plot
fig = plt.figure()
ax = fig.add_subplot(1,1,1)

#make an xy scatter plot
plt.scatter(time_data,voltage_data,color='r', label = 'data')

#label axes
ax.set_xlabel('Time (hr)')
ax.set_ylabel('Voltage (V)')
ax.set_title('Battery Discharge')
plt.legend(loc = 'best')

#save it
plt.savefig('TestData_1.png')
```

This is the script used to calculate the amp-hours for the battery.

```
# -*- coding: utf-8 -*-
"""
Created on Sun Feb 17 20:24:06 2019

@author: maxhu
"""

import numpy as np

R_1 = 100
R_2 = 100

#define data arrays
time_data = []
voltage_data = []
current_data = []
SUM = 0


#read in data
lines = np.loadtxt('Test_1.txt', delimiter=',')
for line in lines:
    time_data.append(line[0] / 3600)
    v_in = ((R_1 + R_2) / (R_1)) * line[1]
    voltage_data.append(v_in)

for n in range(len(voltage_data)):
    I_o = voltage_data[n] / (R_1 + R_2)
    I_f = round(I_o,3)
    current_data.append(I_f)

for n in range(1, len(time_data)):
    A_n = .5 * (current_data[n] + current_data[n-1]) * (np.abs(time_data[n-1]
   - time_data[n]))
    SUM = SUM + A_n

print('The total amp-hours of your battery is:', round(SUM,3), 'Ah')
```