Lab Report 2: Capacitor Discharge With ADC

Max Huggins – UCA Department of Physics Astronomy

February 13, 2019

Abstract

To learn about interfacing with an analog to digital converter chip, a capacitor discharge circuit was made. Discharge data from the capacitor at various resistances was plotted and fit using an exponential decay function.

1 Introduction

The *real* world is continuous, lacking discrete values that can be read into the realm of digital electronics. The Raspberry Pi uses digital logic, 1's and 0's for its operation. To analyze the real world with the Pi, there must be some sort of translation between the analog and discrete signals. To accomplish this, an analog to digital converter (ADC) chip can be used. The specific integrated circuit (IC used for this activity was the ADC0831, which is an 8 bit ADC. The data of interest was the discharge curve of a capacitor at various resistances. The circuit schematic is shown in figure 1. A simple data logging script was written with the general flow shown in 2. The actual script is shown at the end of this report.

2 Analysis

When plotting the data, a decaying exponential function was used for fitting:

$$y = Ae^{-bx} \tag{1}$$

Where the voltage as a function of time of a capacitor discharge takes the form:

$$V(t) = V_o e^{-\frac{t}{RC}} \tag{2}$$

This implies A from (1) is the initial voltage stored in the capacitor and b is given by $\frac{1}{RC}$. For accurate fitting, true capacitor and resistance values were measured with a digital multimeter along with the true Pi output voltage for the amplitude. Values for b were calculated per fit line as shown in the python script for plotting.

3 Experimental Setup

Here are several figures showing schema used for the various activities completed with the finished script also available.

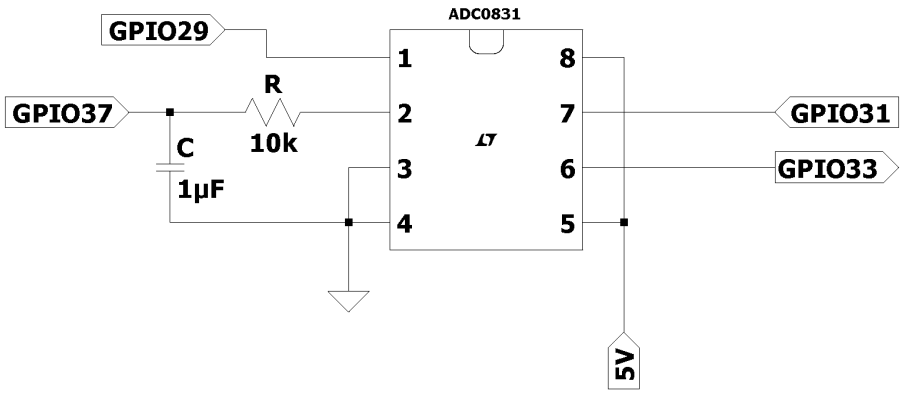

Figure 1: This is the schematic for the ADC capacitor discharge circuit.

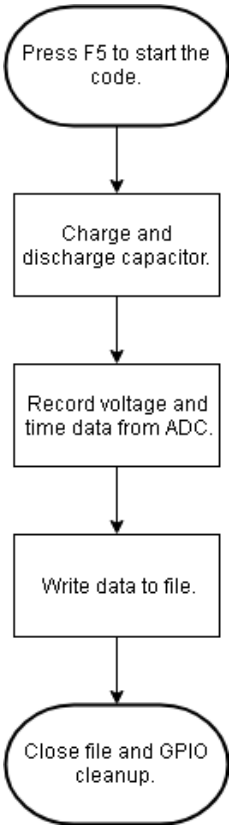

Figure 2: Here is the flow chart for the data logging program.

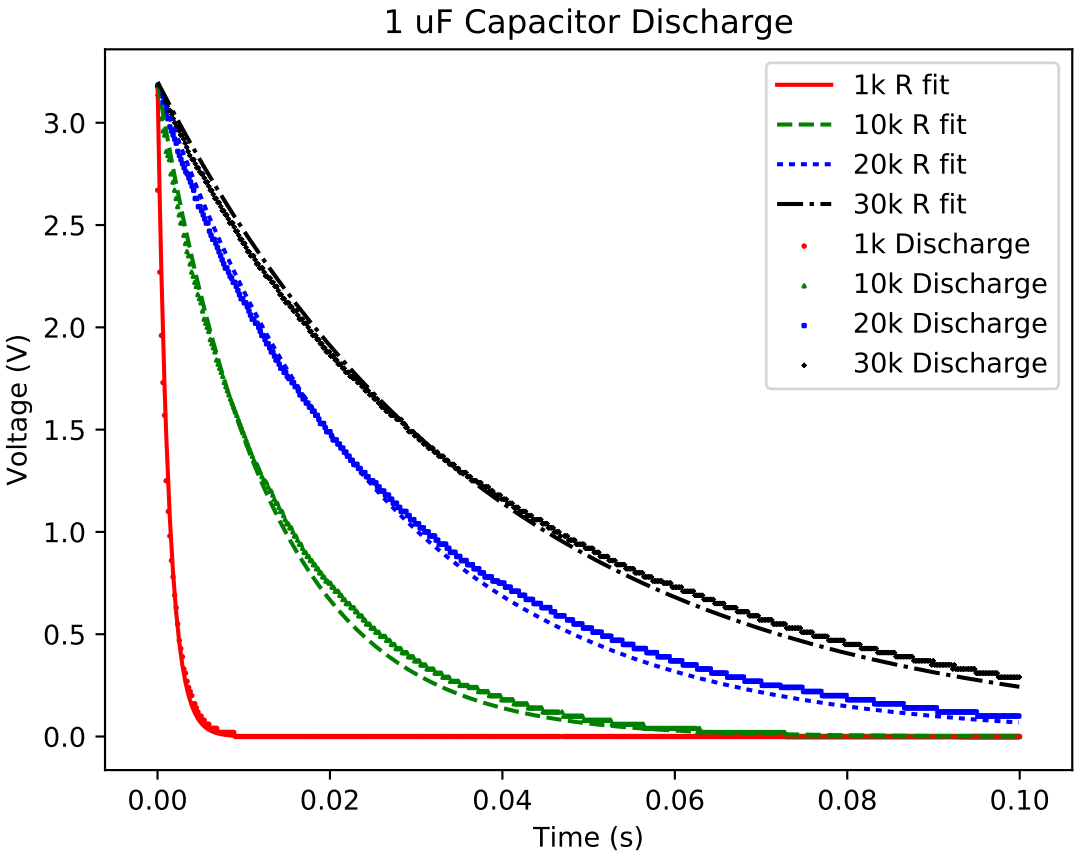

Figure 3: Here is the actual data plotted from the ADC at various resistance values.

4 Conclusion

An analog to digital converter chip was used to take continuous signals from a capacitor discharging and convert them to discrete data signals for the Raspberry Pi. These signals were logged and plotted with a decaying exponential function for the fit. Basics of data logging and analog to digital conversion techniques were used to complete this activity.

Python script for data logging with the ADC.

```
# Max Huggins
# 1/30/19
# Intro to data logging via cap discharge with ADC

import RPi.GPIO as GPIO
import time

#set ADC0831 pins
CS = 29
CLK = 31
DO = 33

#Set capacitor pin
cap = 37

GPIO.setmode(GPIO.BOARD)
GPIO.setup(CS, GPIO.OUT)
GPIO.setup(CLK, GPIO.OUT)
GPIO.setup(DO, GPIO.IN)
GPIO.setup(cap, GPIO.OUT)

#define a function to handle reading ADC data
def readADC():
    #set initial binary to as empty
    d = ''
    #Begin conversation with ADC
    GPIO.output(CS,False)
    #oneclock pulse
    #set CLK low
    GPIO.output(CLK,False)
    #Set clk high
    GPIO.output(CLK,True)
    GPIO.output(CLK,False)
    #end CLK pulse
    #now to read data synced to more clokc pulses
    for n in range(0,8): #read in 8 bits, 0-7
        #one clock pulse
        #set CLK low then high
        GPIO.output(CLK,False)
        GPIO.output(CLK,True)
        GPIO.output(CLK,False)
        DO_state = GPIO.input(DO)
        if DO_state == True:
            d = d+'1'
        else:
            d = d + '0'
        #Do until all bits are read
    #End convo w/ ADC
    GPIO.output(CS,True)
    #Return binary
    return d
#define a function to return voltage
```

```
def calc_volts(d):
    d_int = int(d,2)
    #Pi goes 0-5V and ADC returns values from 0 -> 255 in binary
    volts = 5.0 * d_int / 255
    #Step per binary # is 5V/255lvls
    volts = round(volts,2)
    return volts
try:
    #Create data arrays
    time_data = []
    voltage_data = []
    #run time for experiment
    run_time = .1
    #charging cap
    GPIO.output(cap,True)
    time.sleep(.5)

    my_test = True

    #Discharge cap and record data
    GPIO.output(cap, False)
    start_time = time.time()
    while my_test == True:
        #append time data
        time_data.append(time.time() - start_time)
        #Call function to read a binary data value
        d = readADC() #This returns the binary data
        #append the binary data
        voltage_data.append(d)
        if time.time() - start_time > run_time:
            my_test = False
    #open a data file for writing in the same directory as the working program
    file = open('CapData30k.txt', 'w')
    for n in range(len(time_data)):
        my_volts = 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 all over...")

finally:
    GPIO.cleanup()
```

Python script for plotting ADC data.

```
#Plotting Cap discharge
#max huggins
#1/30/19

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

```
7
8  #define data arrays
9  ONEktime_data = []
10 ONEkvoltage_data = []
11
12 #read in data for 1k resistor discharge
13 lines = np.loadtxt('CapData1k.txt', delimiter=',')
14 for line in lines:
15     ONEktime_data.append(line[0])
16     ONEkvoltage_data.append(line[1])
17
18 #Exponential decay function
19 def my_func(x,a,b):
20     return a*np.exp(-b*x)
21
22 #have to do the following to use the time data in the exp function
23 ONEkfit_time_data = np.array(ONEktime_data)
24 ##########################################################################
25 #define data arrays
26 TENktime_data = []
27 TENkvoltage_data = []
28
29 #read in data 10k resistor discharge
30 lines = np.loadtxt('CapData10k.txt', delimiter=',')
31 for line in lines:
32     TENktime_data.append(line[0])
33     TENkvoltage_data.append(line[1])
34
35 #have to do the following to use the time data in the exp function
36 TENkfit_time_data = np.array(TENktime_data)
37 ##########################################################################
38 #define data arrays
39 TWENTYktime_data = []
40 TWENTYkvoltage_data = []
41
42 #read in data 20k resistor discharge
43 lines = np.loadtxt('CapData20k.txt', delimiter=',')
44 for line in lines:
45     TWENTYktime_data.append(line[0])
46     TWENTYkvoltage_data.append(line[1])
47
48 #have to do the following to use the time data in the exp function
49 TWENTYkfit_time_data = np.array(TWENTYktime_data)
50 ##########################################################################
51 #define data arrays
52 THIRTYktime_data = []
53 THIRTYkvoltage_data = []
54
55 #read in data 30k resistor discharge
56 lines = np.loadtxt('CapData30k.txt', delimiter=',')
57 for line in lines:
58     THIRTYktime_data.append(line[0])
59     THIRTYkvoltage_data.append(line[1])
60
```

```
#have to do the following to use the time data in the exp function
THIRTYkfit_time_data = np.array(THIRTYktime_data)
#######################################################################
#make plot
fig = plt.figure()
ax = fig.add_subplot(1,1,1)

#make an xy scatter plot
plt.scatter(ONEktime_data,ONEkvoltage_data,color='r', label = '1k Discharge',
    s=1, marker = 'o') #For 1k R
plt.scatter(TENktime_data,TENkvoltage_data,color='g', label = '10k Discharge',
     s=1, marker = '^') #For 10k R
plt.scatter(TWENTYktime_data,TWENTYkvoltage_data,color='b', label = '20k
    Discharge', s=1, marker = 's') #For 20k R
plt.scatter(THIRTYktime_data,THIRTYkvoltage_data,color='black', label = '30k
    Discharge', s=1, marker = 'x') #For 30k R
#add the exp curve w/ guesses for the constants a and b
plt.plot(ONEkfit_time_data,my_func(ONEkfit_time_data,3.2,769.2),color = 'r',
    label = '1k R fit', linestyle = '-')#For 1k R
plt.plot(TENkfit_time_data,my_func(TENkfit_time_data,3.2,78.4),color = 'g',
    label = '10k R fit', linestyle = '--')#For 10k R
plt.plot(TWENTYkfit_time_data,my_func(TWENTYkfit_time_data,3.2,38.5),color = '
    b', label = '20k R fit', linestyle = ':')#For 20k R
plt.plot(THIRTYkfit_time_data,my_func(THIRTYkfit_time_data,3.2,25.79),color =
    'black', label = '30k R fit', linestyle = '-.')#For 30k R

#label axes
ax.set_xlabel('Time (s)')
ax.set_ylabel('Voltage (V)')
ax.set_title('1 uF Capacitor Discharge')
plt.legend(loc = 'best')
ax.text(0.4,1.5, 'Exponential decay!')

#save it (I saved it as a pdf to get maximum resolution)
plt.savefig('CapDischargeAll.pdf', format='pdf')
```