|  |  |  |  |
| --- | --- | --- | --- |
| **MEMORANDUM** | | A picture containing graphical user interface  Description automatically generated | |
|  |  | | |
| **To:** | Professor Charlie Refvem, Department of Mechanical Engineering, Cal Poly SLO | | |
|  | [crefvem@calpoly.edu](mailto:crefvem@calpoly.edu) | | |
| **From:** | Jack Butler | |  |
|  | [jbutle10@calpoly.edu](mailto:jbutle10@calpoly.edu) | |  |
| **Date:** | September 26, 2022 | | |
| **RE:** | Lab 0x00 – Serial Communication and ADC Reading | | |
|  |  | | |

**Program Details**

My data collection program begins by importing the pertinent packages, allowing error messages in the callback function, and setting up the hardware and variables. Next, a callback function is defined; this function records the time and voltage if the prescribed number of recordings hasn’t been reached and will reset in preparation for another run once the desired number of recordings has been reached, then print the recordings. At the end of the file are the button handling line and function.

**Data and Analysis**

The program recorded the following data:

Chart

Description automatically generated

Figure 1. Step response of the RC circuit.

Chart

Description automatically generated

Figure 2. Step response of the RC circuit, processed to find time constant.

Figure 2 should have a somewhat linear blue curve, but it clearly does not. This means that the curve shown in Figure 1 does not follow the theorized curve of an RC circuit, so something else must be going on. Although I instinctively want to say there’s a stray capacitance somewhere pulling the voltage down, I probably made some mistake in constructing the circuit. The time constant I calculated was about twice as large as the one given in the notes; back calculating, however, shows this value corresponds to a capacitance of around 10uF, which is within the range given in the handout. This is probably just coincidence, though.

**Source Code**

#import stuff

from pyb import Pin

from pyb import ExtInt

import micropython

#allow error messages in the callback function

micropython.alloc\_emergency\_exception\_buf(100)

#set up hardware

PC0 = Pin(Pin.cpu.C0, mode=Pin.OUT\_PP)

PC1 = Pin(Pin.cpu.C1, mode=Pin.OUT\_PP)

PC1.low()

adc = pyb.ADC(PC0)

tim = pyb.Timer(4,freq=1000)

#set up variables

numsamples = 1500

voltages = numsamples\*[0]

times = numsamples\*[0]

index = 1

#Define callback function; if in time interval, log data, otherwise don't use as callback fn anymore

def measure(cb\_source):

    #making relevant variables global

    global index

    global voltages

    global times

    #if the predetermined amount of time hasn't elapsed yet, then keep sampling - otherwise, turn off the callback fn, set the pin low, and print the recorded data

    if index < numsamples:

        voltages[index] = adc.read()

        times[index] = index

        index += 1

    else:

        index = 0

        tim.callback(None)

        PC1.low()

        print(voltages)

        print(times)

#define what should happen when the button is pressed

def buttonhit(button\_source):

    tim.callback(measure)

    PC1.high()

#setting up the button function

button\_int = ExtInt(Pin.cpu.C13,ExtInt.IRQ\_FALLING,Pin.PULL\_NONE,buttonhit)