MEMORANDUM

To: Charlie Refvem, Department of Mechanical Engineering, Cal Poly SLO
crefvem@calpoly.edu

From: Caiden Bonney
clbonney@calpoly.edu

Antonio Ventimiglia
ventimig@calpoly.edu

Date: 10/2/2025

RE: **Interrupt Callbacks and ADC Reading**

Utilizing the timer interrupt callbacks and analog-to-digital (ADC) functionalities on an STM32 Nucleo board, the step response of a simple RC circuit was recorded. This response was then compared to the theoretical step response of the RC circuit using the measured values of the components and simplifying assumptions to lower modeling complexity. With a 3.53% error between the experimental and theoretical time constants, it was concluded that the tolerances on the electrical components present a larger potential source of error than the internal dynamics of the Nucleo.

The RC circuit connected to the Nucleo board is illustrated in Figure 1, where the input voltage is provided by pin C1 of the CN7 Morpho header, the ADC reading is performed by pin C0 of the CN7 Morpho header, and all component values are nominal.

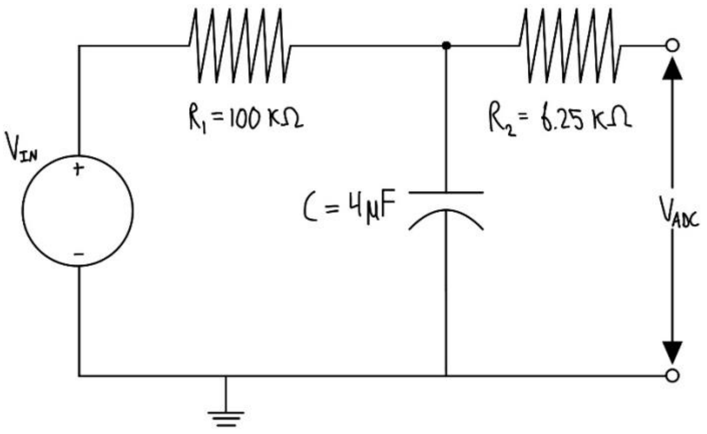

Figure 1. RC Circuit Diagram with Nominal Values

To accurately determine the theoretical behavior of the system, the electrical components shown in Figure 1 were measured using a multimeter, and the recorded values are presented in Table 1.

Table 1. Nominal and Measured Electrical Component Values

	Nominal	Measured
R_1 [kΩ]	100	99.6
R_2 [kΩ]	6.25	6.28
C [μF]	4	4.28

Using the values from Table 1, the estimated time constant was calculated: $\tau_{est} = R_1 C = 0.426$ sec.

The team's Python script was then compiled and flashed onto the Nucleo and ran. The voltage step response recorded by the Nucleo's ADC and the theoretical waveform can be found in Figure 2. The theoretical curve uses the steady-state voltage, V_{ss}, and the experimental time constant, τ_{exp}, which is calculated later in the report.

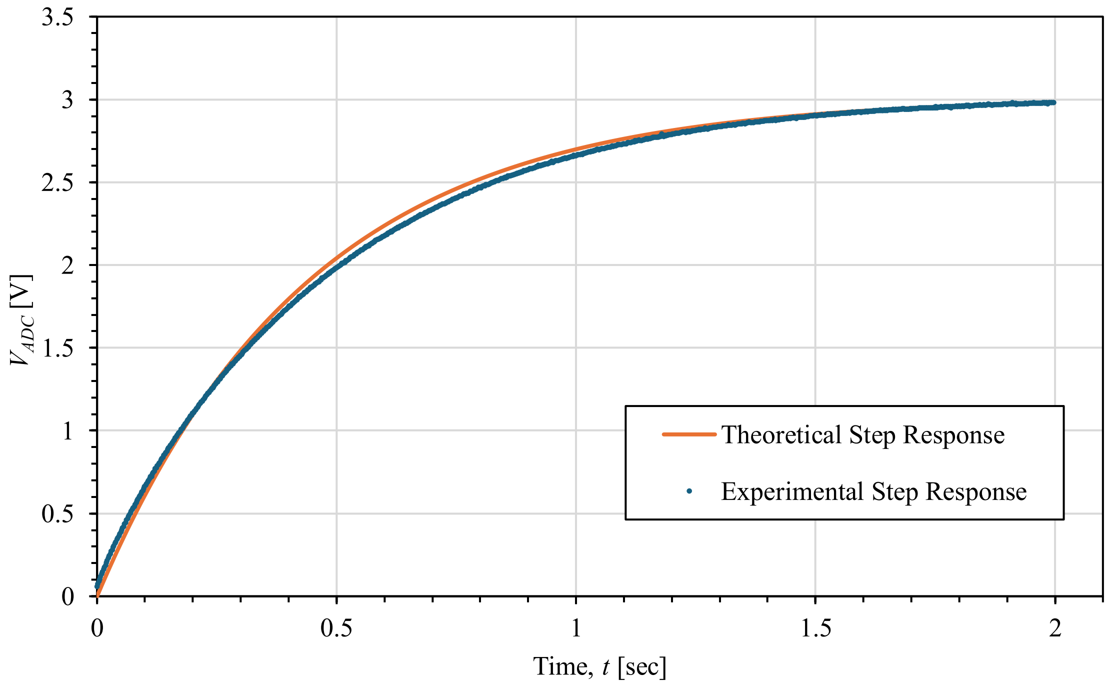

Figure 2. Experimental and Theoretical Step Response of RC Circuit

Since the theoretical step response is constructed from the experimentally determined steady-state voltage and time constant, Figure 2 is a qualitative comparison between the two curves and their general behavior. It can be noted that the curves exhibit remarkable conformity, providing the team with confidence that the first-order approximation used to model the system's behavior is adequate in describing the system.

To determine the experimental time constant, the theoretical voltage relationship for the step response in Equation 1 can be manipulated.

$$V_{ADC} = V_{IN}\left(1 - e^{\frac{-t}{\tau_{exp}}}\right) \tag{1}$$

Where V_{ADC} is the voltage across the ADC pin, V_{IN} is the theoretical voltage source that V_{ADC} asymptotically approaches, τ_{exp} is the system's experimental time constant, and t is time. We can linearize the curve by rearranging Equation 1 into Equation 2:

$$\ln\left(1 - \frac{V_{ADC}}{V_{IN}}\right) = -\frac{1}{\tau_{exp}}t \tag{2}$$

We can see from Equation 2 that the slope of the linearized curve is equal to the negative inverse of the experimental time constant. By plotting the data from Figure 2 according to Equation 2, Figure 3 provides the experimental time constant for the system.

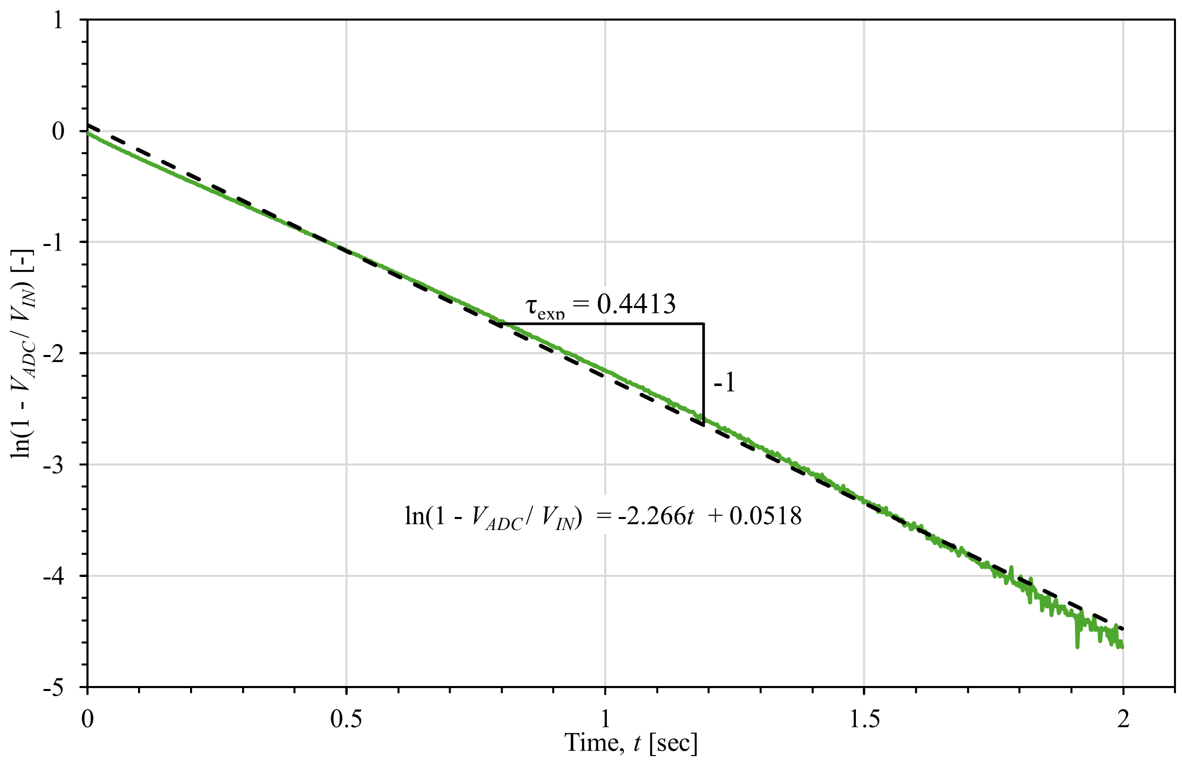

Figure 3. Linearized Step Response

From Figure 3, the experimental time constant is determined to be $\tau_{exp} = 0.4413$s. The percentage error of τ_{exp} from τ_{est} is calculated according to Equation 3

$$\%\ Error = \frac{\tau_{exp} - \tau_{est}}{\tau_{est}} \text{X } 100 \tag{3}$$

This yielded a percentage error of 3.53%.

Given the low error, the team concluded that a first-order RC system can accurately model the step response measured by the Nucleo's ADC. The time constant can be approximated as $\tau = R_{exp}C_{exp}$. The internal mechanics of the Nucleo do not significantly affect the overall system behavior, especially when considering that component tolerances alone could introduce a much larger error if not properly accounted for.

References:

- Refvem, C. (2025). *Lab 0x00 Interrupt Callbacks and ADC Reading* [Unpublished lab instructions]. Department of Mechanical Engineering, Cal Poly. Retrieved from https://canvas.calpoly.edu/courses/161863/assignments/1368600?module_item_id=4790186
- Refvem, C. (2025). *Lab 0x00 RC Circuit Analysis* [Unpublished lecture notes]. Department of Mechanical Engineering, Cal Poly.

Code:

```
from pyb import ADC, Pin, Timer, ExtInt
from array import array
from time import sleep_ms

# Initializing variables
data = array('H', (0 for _ in range(1000)))
idx = 0
done = False
waiting = False
button = False

# Function definition
def tim_cb(tim):
    global data, idx, done
    if idx >= 1000:
        tim7.callback(None)  # disable the callback
        done = True
    else:
        data[idx] = adc.read()
        idx = idx + 1

def button_pressed(pin=None):
    global button
    button = True

def start_step():
    global done, waiting, idx
    if not waiting:
        # Print headers
        print("idx, val")

        # Ensure pin starts low
        PC1.low()
        sleep_ms(5000)  # allow pin to settle

        # Reset index and done flag
        idx = 0
        done = False

        # Start timer callback first
        tim7.callback(tim_cb)

        # Take one sample before stepping high
        data[0] = adc.read()
        idx = 1

        # Step pin high
        PC1.high()

        wait_and_print()
```

```
def wait_and_print():
    global data, idx, done, waiting, button
    waiting = True
    # Wait for data collection
    while not done:
        pass

    # Print data collected
    for i, val in enumerate(data):
        print(f"{i}, {val}")
    waiting = False
    button = False

# defining Power Pin
PC1 = Pin(Pin.cpu.C1, mode=Pin.OUT_PP)  # Power
PC1.low()

# defining Output Pin
PC0 = Pin(Pin.cpu.C0, mode=Pin.ANALOG)  # Output
adc = pyb.ADC(PC0)

# define timer
tim7 = Timer(7, freq=500)  # create the timer object
tim7.callback(tim_cb)  # assign the callback

# Creating button
button_int = ExtInt(Pin.cpu.C13, ExtInt.IRQ_FALLING, Pin.PULL_NONE, button_pressed)

while True:
    if button:
        start_step()
```