Name: ____________________

Collaborators: ____________________

Thayer School of Engineering • DARTMOUTH

ENGS 28, W26

Lab 2 — Simple Digital Inputs

Due: Tuesday January 20, 2026 2:15 PM

Contents

Deliverables

1 Overview

1.1 Place in course

During class and in Lab 1, you learned a bit about how to control a simple digital output from your microcontroller. In this lab, you will develop the ability to read a simple sensor and act upon it's value.

1.2 Learning Objectives

- Analog electronics: How an RC circuit can be used to debounce a pushbutton switch.
- Using microcontroller pins for input as well as output.
- Microcontroller programming: state machine design pattern.
- Microcontroller programming: handling real-time inputs via polling.

1.3 Equipment

- Nucleo + breadboard
- AD2 with small flying lead bundle
- Three LEDs, with 220-ohm current limiting resistors[1]
- Two pushbutton switches
- Two 0.15 microfarad (µF) capacitors and two 0.022µF capacitors

2 Technical Study

Review the readings and class notes for Days 1-4.

Work the prelab problems below. Neatly write up your solutions in the spaces provided (add more pages as needed). Upload your answers to Canvas along with any code you write for the design challenges. When you submit work for this class, your goal is to communicate clearly to the reader that you understand the solution. Invest time to organize your thoughts in your submission and always show your work. Double, triple check your submission to make sure that your work scanned properly and that you have answered all of the problems before submitting (you don't want your work to be illegible, upside-down or missing problem 3!).

[1] 200Ω are fine too

2.1 RC Debouncer Design

We saw in class how a capacitor can be used to debounce a pushbutton switch. Figure 1 shows this technique in the context of our I/O driver.

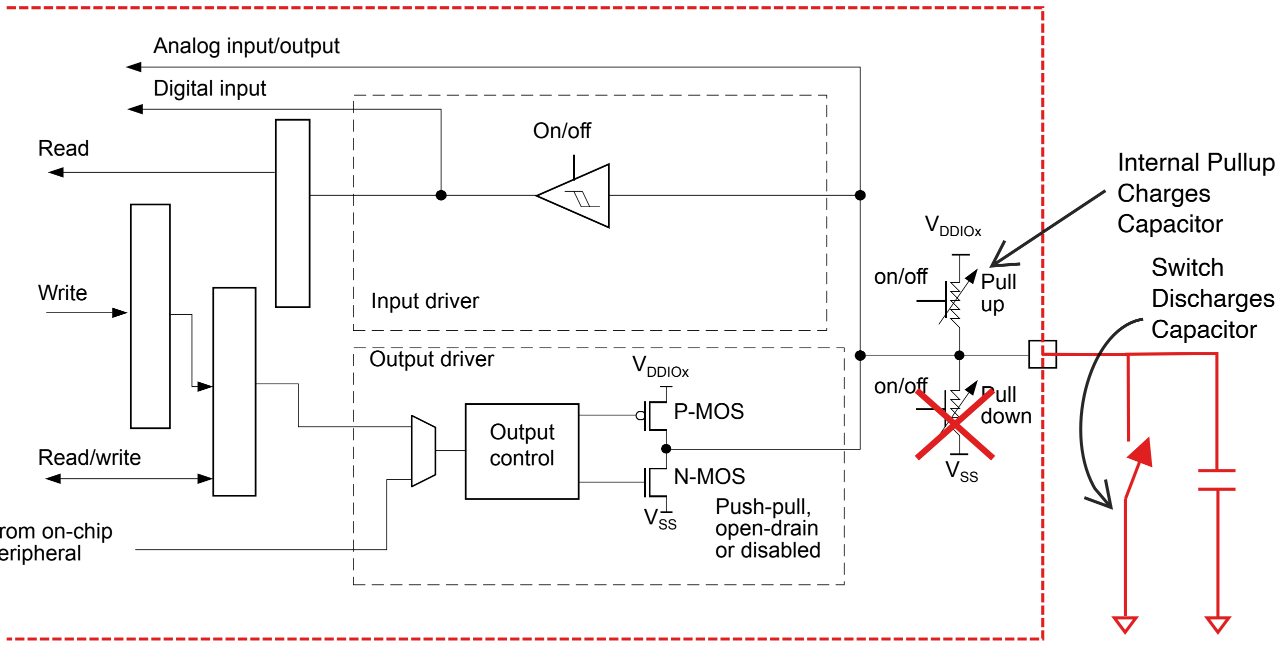

Figure 1: An RC Circuit Used to Debounce in Input

In order to properly choose the capacitor value, it is necessary to understand how the debouncer works. A simplified circuit, with the microcontroller components removed, is shown in Figure 2. As you examine the two figures, think about how the simplified figure is synthesized from the figure in the datasheet.

Recall that a capacitor stores electric charge (Q), and the voltage across the capacitor, $V_C = Q/C$. When the switch is closed, any charge in the capacitor flows through the switch to ground. Because the switch has negligible resistance, this discharge is may be considered to be instantaneous. Current from the power supply (V_{DD}) also flows through the switch to ground, limited by the pullup resistor to a reasonably small value.

Deliverable 1: Datasheet *(1 point)*

Where in the STM32C031 data sheet are the values of R_{pu} and V_{IH} specified? Use the STM32C031x4/x6 datasheet found on Canvas. The file is called `stm32c031c6.pdf`. Specify the NAME of the table in which these values are found, as well as the page number. While not always true, most datasheets will have a table of the same name containing similar values. This is a good place to look when questioning compatibility between parts you might be considering for a design.

Name of the table : ______________________

R_{pu}: ______________________ Page: ______________________

V_{IH} : ______________________ Page: ______________________

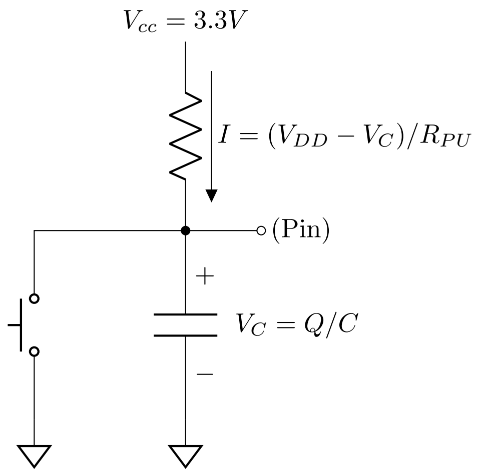

Figure 2: Simplified RC circuit

Deliverable 2: Initial Current *(1 point)*

Assuming the minimum value of $R_{pu} = 25K\Omega$ (which allows the most current to flow), what is the current I when the switch is closed?

When the switch opens, the current from the power supply flows into the capacitor, depositing charge and causing the voltage V_C to increase. The capacitor voltage in an RC circuit follows the "saturating exponential" curve shown in Figure 3.

Initially, when the capacitor is discharged, the current flow is at its highest value, which you calculated in Deliverable 2. Because current is the rate of change of charge flow, dQ/dt, and the capacitor voltage V_C is proportional to Q, the rate of change (slope) of V_C is highest at $t = 0$, as you can see on the graph. As the voltage on the capacitor increases, the current decreases, according to the equation $I = (V_{DD} - V_C)/R_{PU}$. The lower current charges the capacitor more slowly, as you see from the decreasing slope of the curve. When the capacitor is nearly fully charged, hardly any current flows, and asymptotically, $I \to 0$ as $V_C \to V_{DD}$.

The approximate time to fully charge a capacitor is $5RC$. A large capacitor charges more slowly than a small one for a fixed value of R.

Here is how the capacitor helps to debounce the switch. When the switch is closed, the capacitor discharges virtually instantaneously, producing a LOW ($V_C = 0$) at the pin. When the switch bounces open, current through the pullup resistor begins to charge the capacitor back up. If the capacitor voltage reaches V_{IH}, then the pin will see a spurious HIGH. When the switch bounces closed again, there will be another LOW, etc. Your task in designing the debouncer is to choose the capacitor large enough so that it cannot charge up to VIH during the bounce and produce the spurious HIGH voltage at the pin.

The mathematical expression for the voltage in this circuit

$$V_C(t) = V_{DD}\left(1 - e^{-t/RC}\right) \tag{1}$$

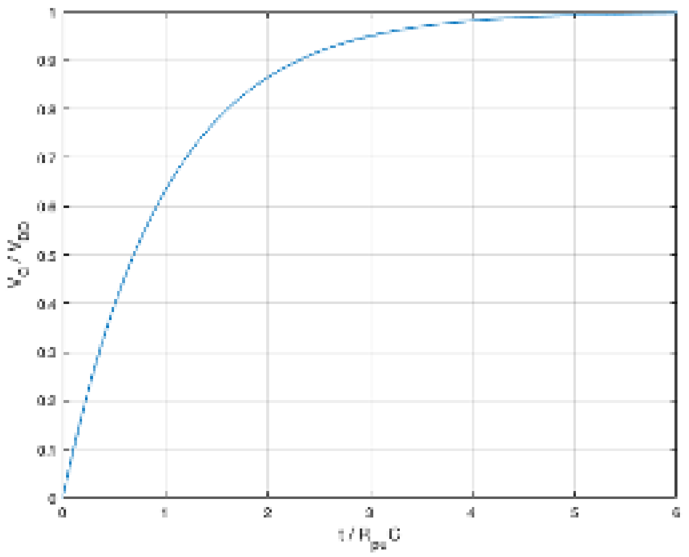

Figure 3: A saturating exponential

We need to know when the voltage would reach V_{IH}.
Deliverable 3: Time Constant *(2 points)*

Derive an expression for t (in terms of V_{DD}, V_{IH}, Rpu, and C) such that $V_C(t) = V_{IH}$. **Show your work**.

Deliverable 4: Effect of Capacitance on τ *(1 point)*

What does this expression predict if a larger capacitor is used: Does it go up or down, and does this make sense physically? Explain.

Deliverable 5: Debounce Design *(2 points)*

Previously, you have measured the bounce time of your switch with the oscilloscope. You know $V_{DD} = 3.3V$, and from the data sheet, the value of V_{IH} and (worst case) $R_{pu} = 25K\Omega$. What do you get for a value of the capacitor so that the time t in the above equation is longer than the bounce time? (That is, the capacitor voltage does not reach V_{IH}.) How does it compare to the 0.15 µF value we used in class? (Show your work)

Deliverable 6: Cap Testing *(1 point)*

Use your test circuit from Tuesday's homework, with the LED toggle code and the AD2. Compare the glitchiness of the circuit with these three capacitor values: $0.022\mu F$ (from your kit), $0.15\mu F$, and $0.3\mu F$ (two $0.15\mu F$ capacitors in parallel). Discuss what you see.

Deliverable 7: Choice of R *(2 points)*

The data sheet says that R_{pu} can vary from $25K\Omega$ to $55K\Omega$. We have called $25K\Omega$ the worst-case value for this circuit. Why is this, rather than $55K\Omega$, the worst-case value?

3 Design Challenges

3.1 Pushbutton controlled counter

In your Lab 2 directory, make a copy of your blinkyCNT.c code from Lab 1 and modify it so that a button press will cause the counter to pause, and a second press will cause it to resume. (You may have already completed this).

Deliverable 8: Pushbutton Counter *(2 points)*

> Obtain a signature or submit a video of your pushbutton counter. Upload your code to Canvas.

TA or Instructor Signature ____________________

3.2 Reaction Time Game

Design and implement the two-person reaction time game demonstrated in the lab videos on Canvas. Each player has a (debounced) button and a red "WIN" LED. A third (green) LED serves as the start signal. Here are the specifications for the gameplay:

1. Initially, the starter LED is ON, or perhaps blinking rapidly.
2. Players can press either button to start the game.
3. The starter LED "counts down" with three flashes, one second apart, then glows steadily.
4. The players press their buttons as soon as they see the starter LED become steady. The player who presses their button first wins, and their LED lights (or blinks rapidly) for a second.
5. The system returns to step 1.

You can reuse the three LEDs from your Lab 1 setup, and just add the two switches to your breadboard.

Break the problem down into small steps that can be individually built and tested. Your debugging tools are limited (e.g., flash an LED to indicate that a particular portion of the code has been reached). Therefore, being slow and careful will be the best strategy.

Here is a strategy hint to get you started. The gameplay as outlined above has five steps. In each step, certain actions are being taken (e.g., light an LED), and certain events are being watched for (e.g., a button press). When an event is detected, additional actions may be taken, and the game moves to the next step. It makes sense to organize your code in a similar way. At the beginning of each lap around the while(1) loop, check which step you're on and execute the code particular to that step. When you reach the end of the loop, you will either be at the same step (because no event was detected), or be poised to enter the next step (because an event was detected). In the language of digital systems, these steps are called states.

Start with pencil and paper (and not with an editor). Map out the different states that your game will go through and what events cause a transition from one step to another. Once you have all of the game logic mapped out, write pseudo-code (on paper, with a pencil). Only once all these are done should you open an actual editor on your computer.

Remember to use the fixed-size data types `uint8_t`, `uint16_t`, etc, rather than char, short, int, etc. Be sure to document your code, including comments saying which pins you use for input and output, so that someone else could reproduce your setup. It is an adage in software development that code is written once, but read many times (by humans who have to debug and maintain it). Always strive to write readable code.

Deliverable 9: Reaction Game *(8 points)*

> Obtain a signature or submit a video of your reaction game in action. Upload your code to Canvas. In addition to your code, include your map of the game logic, pseudo-code, and a diary of your design and debugging process.

TA or Instructor Signature ____________________

3.3 Optional

Optional features for extra challenge and personal satisfaction (but no additional points)

1. A false start, pressing one's button out of turn, awards a win to the other player.
2. With practice, a player can time the flashes and synchronize their button presses. To discourage such behavior, make the delay between the last flash and the steady ON of the starter LED a little bit random. One good way to do this is to run a fast counter "in the background", and when a button is pressed to start the game, the value of the counter is grabbed and used to set the delay between the last starter LED flash and the steady LED state. It will be perceived as random by the players, without the computational overhead of an actual random number generator.

Deliverable 10: Reaction Time optional features *(0 points)*

Obtain a signature or submit a video of your extra features. Upload your code to Canvas.

TA or Instructor Signature ____________________