TABLE OF CONTENTS

1 LAB OBJECTIVES

- Introduction to lab equipment and microcontroller
- Understand how we might use them to control digital outputs.
- Understand the limitations switch polling
- Using timers and interrupts to mitigate the limitations of switch polling
- Understand the potential problem (e.g., race conditions, shared mutable variables, etc) and potential solutions to these problem.

2 BACKGROUND

The purpose of this lab is to introduce students to using micro-controller for basic real-time (RT) system and the basic tools used by embedded system engineers: oscilloscope and function generator. The oscilloscope used to troubleshoot electronic systems. Additionally, real-embedded system signals are too complex (e.g., sensor data) and difficult to to analyze. The function generator is used to generate electronic signals with specific known characteristics, thereby enabling an engineer to test and examine a circuit. Finally, you will use a micro-controller to implement specification of a simple interrupt controlled system.

3 LOGISTICS

- This lab is meant to review knowledge and skills that you acquired in your previous courses and to make sure that all students have the same common understand, bridge knowledge and skill gaps, if any, and prepare students to the future labs.
- The labs are individuals. Each students shall be able to setup his/her own circuitry and be comfortable in using the troubleshooting tools that are available in the lab.
- Although you're free to use any micro controller of your choice, I recommend you use an AVR based MCU (eg., MCU in Arduino UNO, Nano and Mega).
- In the upcoming labs, we will use an more suitable MCU
- Please read this lab throughout before you ask questions.

4 LAB EXERCISES

4.1 Exercise 1—switch bouncing

4.1.1 Background

In embedded systems, an important—but often overlooked—concern s is the problem of switch bounce. The basic mechanisms involved are the mechanical design of the switch and the large electric fields that develop between switch contacts that are very close to each other (almost making contact with each other but not quite making contact). What results from this situation is arcing between the switch contacts until they finally make permanent contact with each other. Unless mitigated, this bouncing can cause undesirable results.

In this exercise, you will use an oscilloscope to measure the noise that occurs when a mechanical switch change state.

4.1.2 Lab circuitry

In this exercise, you will build and program a 4-bit LED binary counter. The counter counts from 0 to 15 (0000 to 1111) using LEDs (e.g., a digital 5 is 0101, which implies that LED1 and LED3 are ON, and other LEDs are OFF). Four LEDs and two switches are connected to micro-controller (MCU):

- LED 1 (D1) is connected to GPIO #1 of the MCU
- LED 2 (D2) is connected to GPIO #2 of the MCU
- LED 3 (D3) is connected to GPIO #3 of the MCU
- LED 4 (D4) is connected to GPIO #4 of the MCU
- A push button switch (SW1) is connected to GPIO #5 of the MCU
- A push button switch (SW2) is connected to GPIO #6 of the MCU

4.1.3 Lab instructions

- Draw a schematic for how you would connect a switch to your MCU. The schematic should show the switch, the MCU input pin, and the LEDs any other components (e.g., resistors) connected to the MCU.
- Ask the lab technician to give you a breadboard, two switches (preferably single-pole, double-throw (SPDT) pushbuttons if available) , 4 LEDs and 6 220-ohm (any value

less than 1k ohm is also acceptable) pull-up resistors and a MCU board of your choice (Arduino, ESP32, ESP8266 or any MCU you are comfortable with).
- You may also needs some jumper wires to connect your actuators to the MCU boards. Please ask for these if necessary.
- Connect the two switches and the 4 LEDs to the micro controller as discussed in Section 4.1.2. For consistency, make sure that ALL resistors are connected to the switches in a pull-up configuration (pull-down would work fine as well but you should not use them for this lab). Also,note that each LEDs should be connected to a 220-ohm.
- Write a program that counts how many times a switch is pressed. The program should work as follows:
  - At the beginning, all LEDs are OFF and the counter is zero
  - If switch SW1 is pressed, the counter is increased by one, and the value is displayed using the 4 LEDs
  - If switch SW2 is pressed, the counter is reduced by one, and the value is displayed using the 4 LEDs
  - If the exceeds 15 or is less than 0, the counter is reset back to zero
- Plug an oscilloscope probe into channel 1 of the oscilloscope, and connect the probe to the MCU pin connected to the switch (You can use an alligator clips and wires if needed)
- You should make sure channel 1 on the oscilloscope is on, and channel 2 is off . (The light for channel 1 should be lit, and the light for channel 2 should not)
- Set up the oscilloscope to inspect what happens when the switch is pressed. You'll need to adjust the vertical scale and offset for channel 1 to do this. Adjust the time scale about 5 ms (the exact value doesn't matter, as long as it's around that range),and the time o set to zero. **If you have any question on how to use the oscilloscope, please contact the lab technician.**
- You may also prefer to set the oscilloscope trigger when the button is pressed. For instance, you can set an event trigger when the voltage crosses 2V in either direction. In this way, you can see the exact moment when the switch is pressed and released.
- Describe what the oscilloscope display. Take a screenshot of the display (the oscilloscope normally provide a mechanism to take a screen shoot. Please use it instead of using your phone).
- Is the transition from ON to OFF look as you expected?
- Zoom in horizontally on your oscilloscope and adjust the time scale to about 100 µs and observe the trace on the oscilloscope when the switch is pressed. On this time scale, does it look like a clean transition you expected?
- Write down all your observation in your report.

4.2 Exercise 2—software switch de-bouncing

4.2.1 Background

There exist several methods to de-bounce a switch[1]. For instance, a simple RC debouncer (Figure 1) can be used used to reduce the effect of a bouncing switch.

4.2.2 Lab instructions

- Repeat the lab instruction described in Section 4.1.3 and use the circuit in Figure 1 to debounce each switch.

[1] https://my.eng.utah.edu/~cs5780/debouncing.pdf

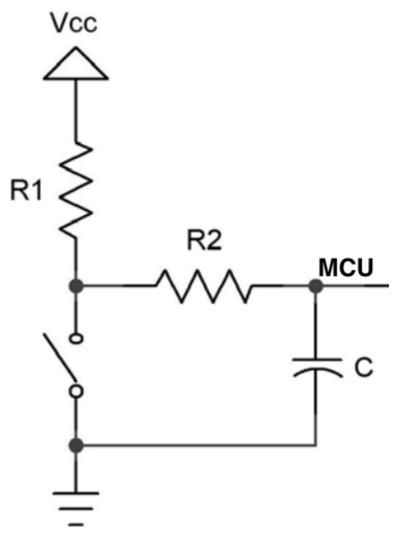

FIGURE 1. An RC debouncing circuit

- Use a capacitor of $C = 0.1\,\mu\text{F}$ (or any other available capacitor but a bigger capacitor is better) and use the resistors of $R_1 = R_2 = 200\,\Omega$
- Connect the output of the switch to the oscilloscope and describe your observation. Does it look like a clean transition you expected?
- Write your observation in your report.

4.3 Exercise 3—blocking polling

In this exercise, you're asked use the circuit described in Section 4.1.2 and to write a program that read the digital input. Your program should work as follows:

- All LEDs are OFF at the start of the program
- Once the program begins, LED1 and LED4 blink continuously at 1 Hz
- While the switch SW1 is pressed, LED2 is turned ON and stays ON until the switch SW1 is released.
- When SW1 is pressed and then released, turn ON LED3 for one second, and turn it OFF

To complete this exercise, you should do the following:

- Sketch a flowchart of the program
- Write, debug and simulate the program using Proteus VSM
- Does your program work as intended? If not, describe the problem, if any, and explain why this is the case.

NOTE—At the end of this exercise, you should note that:

- While the digital input SW1 is pressed, this code blocks the CPU in a tight polling loop. This is also known as spinning.
- Switch pooling is not power or CPU efficient.
- Once the switch is pressed, the CPU is blocked and cannot do anything else.
- While it is not always bad practice to monitor micro-controller peripheral hardware this way (especially when the polling duration is short), this is highly discouraged is several case because the switch presses have an indeterminate time. Thus, spinning becomes wasteful of CPU time and most of all, power.

4.4 Exercise 4—non-blocking polling

NOTE: Although switch polling is simple, it has several disadvantages:

- It uses a lot of CPU resource even when there are no changes in the inputs. In reality, most modern embedded systems are often idle and polling would unnecessary drain their battery.
- The loop cycle time is variable (e.g. due to conditional statements). Therefore the sampling interval of the inputs is not constant and is said to jitter.
- If the loop cycle is ever too long, input changes could be missed and data lost. The delay gets longer as more code is added.

This exercise aims to address some of these issues. It uses timers to create non-blocking delays and interrupts to asynchronously detect when a switch is pressed and released.

4.4.1 Reading materials and exercise

before attempting the lab, please review the lecture notes on interrupts and timers. You should also read the following online resources:

- Read to review the basics of AVR timer
- Read this online resource to understand the basics of AVR interrupts
- Read this online resource to understand external hardware interrupts in AVR ATmega32

4.4.2 Lab instructions

Complete the previously completed exercise using interrupts and and timers. It is helpful to keep the following in mind:

- To detect whether a switch has been pressed or not, please use external interrupts. It is recommended to revised discussion in the lecture about this topic. You should also read this online resource to understand external hardware interrupts in AVR ATmega32
- From your readings and the discussion in the class, it should be clear that that not all pin support external interrupts. Thus, you should re-arrange the connection of the two push button SW1 and SW2 to take advantage of external interrupts.
- You should not use the built-in delay $\_delay_m s()$ function when creating delays. Instead, you need to create no-blocking delays using a timer. Please review the lecture slides and read to review the basics of AVR timer to understand how to create a delay with a timer.

5 LAB REPORT INSTRUCTIONS

Each exercise should be accompanied by **1 one page** lab report. The lab report should be written as if the reader was another member of the class. With your report in hand, one should be able to reproduce what you did and understand it well enough to add to it. The report shall contain the following sections

- **Abstract** —1 paragraph describing what you did. It should explain the purpose of the lab, results, its scope, summary of your solution and a conclusion.
- **Introduction** —What is the lab about project? Why is it interesting? What technical details does your reader need to understand?

- **Circuit description** Draw block diagram of the circuit. Please note that you should note use the Proteus diagram. Instead, you should manually draw a separate block diagram (and not an actual circuit) dthat shows only components and pins that are being used on the MCU. Discuss how various resistors, transistors have been selected. Show any pertinent calculation when necessary.
- **Code description** —Start with a very high level description of your code (block diagrams, perhaps) and how it uses subsystems... Get increasingly detailed in your description until you get to the level of short sections of code that are critical to the proper operation of your project.
- **Results and discussion**
  - Document the procedure of the lab, written in first person narrative, documents how the experiment occurred so that someone else would be repeat your results.
  - Clear description of your observation when testing your systems
  - Clear circuit diagrams should be included
  - Document your results are clearly stated and provide evidence (e.g., screenshots of the oscilloscope)
  - Figures, equations, and tables should be labeled and formatted according to engineering standards.
  - In your observation, did the system work as expected? If something did not work as expected, what do you think is the reason? How could your solution be improved?
- **Conclusion** —

6 GRADING CRITERIA & DISTRIBUTION

- **FUNCTIONALITY** **50%**
  - Each exercise works as expected 20%
  - The student can use an oscilloscope properly 10%
  - The student understands his/her code and has the ability to explain and debug the code systematically 10%
  - The student can answer any tangential questions asked 10%

- **CODE QUALITY** **20%**
  - Proper comments (proper and no misleading or useless comments) 5%
  - Proper coding coding style 5%
  - Proper code organization and modularization 5%
  - Does not violates major embedded C coding standard 5%

- **LAB REPORT** **30%**
  - introduction and abstract 5%
  - accurate description of results—clear, concise but complete 5%
  - clear circuit diagrams 5%
  - clear software description 5%
  - clear and concise conclusion 5%
  - Clarity in writing 5%

- **CHEATING** Any cheating will forfeit your grade. Cheating includes:
  - Copying the work of other students in whole or in part

  - Stealing another student's work and submitting it as one's own work
  - Turning in a lab report not done by you.
  - Allowing the copying of the write-up of your laboratory work or homework by others

7 LAB SUBMISSION

- All exercises are individual and shall be submitted individually
- The lab shall be submitted no later than 23:59 on Sunday, November 6, 2022
- You should submit the report, all your code file (c or cpp files) in one .zip file and submit them through the e-learning platform before the deadline
- Only work submitted through the e-learning platform will be graded. I will not accept any submission through email.
- Please submit your work before the deadline. I will not accept any submission through my email no matter the reasons
- In-lab verification of you work will be done on Monday November 7, 2022 at 8:00. I will check that the hardware and software work as expected.
- **Save your code for each exercise separated for easy verification.**
- **Late submissions are not accepted after the deadline**.
- If you're not available in the lab, you will get a zero for the assignment (unless you can provide a documented evidence for your absence)