# ECSE 421 - Lab 3: Interrupts & I/O

## Exercise 1: A Simple Doorbell

**Task**: Write a program that rings the buzzer every time the button is pressed, using an interrupt and ISR to control the buzzer. Implement a two-tone buzz (or feel free to get more creative with your chime!).

- ➔ DEMO: E1.MOV
  - https://drive.google.com/file/d/1kst7-a4fh7jLpxHXHepnI1-70DpDUu8K/view?usp=drive_link
- ➔ BONUS (Just for fun): E1_Bonus.MOV
  - https://drive.google.com/file/d/1DYZLTrL5RAISeo7sUTQc7AtBeitJxRZ6/view?usp=sharing

**Question**: Will the `delay()` function work inside the ISR (interrupt service routine)? What about `micros()`? It may be helpful to try this out. Give an explanation for your answers.

- ➔ **ANSWER:** It is mentioned in the `attachInterrupt` documentation https://www.arduino.cc/reference/cs/language/functions/external-interrupts/attachinterrupt/ that:

  > **Notes and Warnings**
  >
  > Inside the attached function, `delay()` won't work and the value returned by `millis()` will not increment. Serial data received while in the function may be lost. You should declare as volatile any variables that you modify within the attached function. See the section on ISRs below for more information.

With some testing, I observed that `micros()` and `millis()` print out the time of the interrupt event (I had `CHANGE` as the interrupt trigger), without slowing down due to the tone or the push button. `delay()` also has no effect regardless of how long the wait duration is

# Exercise 2: Your Rotary Encoder, Enhanced!

## Direct accessing registers

**Task 1:** Write a program that outputs a counter on the serial monitor.

- ➔ DEMO: E2_T1.MOV
  - https://drive.google.com/file/d/1AMuSZF-SQ9CkmsR5Crd49LgfkzK-F4Sv/view?usp=drive_link

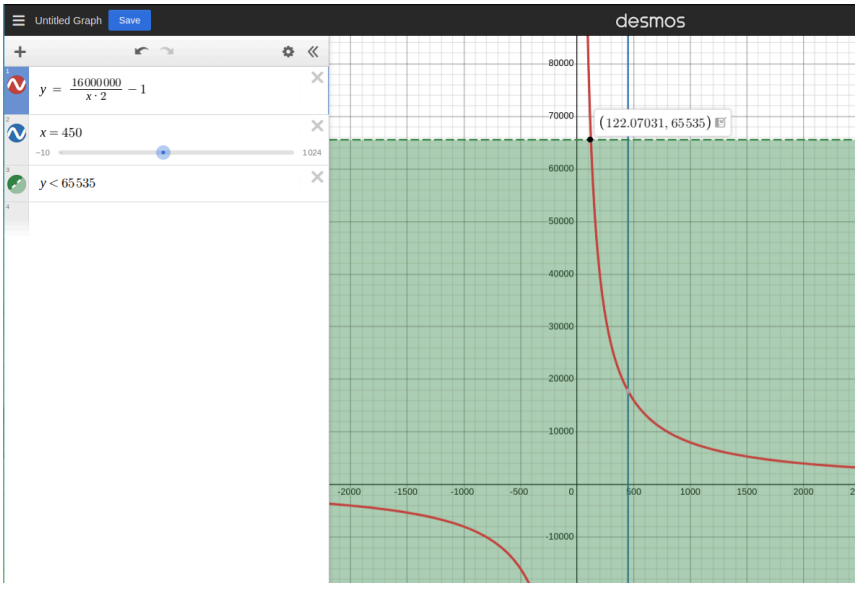
## Generating a PWM signal

**Question 1:** Let's say you want an interrupt to occur every 500 ms. Assuming you're using Timer1, what valid combination of OCRx value and prescaler value would allow the interrupt to occur at the desired frequency? Show your calculations.

- ➔ **ANSWER:**
  - Timer1 is a 16-bit timer, meaning it can count up to 0xFFFF (65535).
  - Interrupt frequency = $f_{interrupt} = 1/500ms = 2Hz$
  - Knowing that the Arduino operates on 16MHz (source: Arduino Forum)
  - Knowing that Timer1 in CTC mode, meaning we can calculate OCRx and prescaler value from equation

$$OCRx = f_{CPU} / (prescaler * f_{interrupt}) - 1$$

  - Plotting OCRx as a function of prescaler, we can get:




Based on the graph, we can pick any value of prescaler above 123. We would want a lower prescaler since it can provide higher resolution timing and less numerical errors from rounding. We also want the prescaler to be a power of 2.

➔ Hence, our best choice is Prescaler = 256 ($2^8$), which leads to OCRx = 31 249
➔ NOTE: the answer was partially supported by explanation of Arduino AVR Timer, OCR, CTC, and prescaler from ChatGPT 4.0.

**Task 2**: Using one of the timers for the PWM and the rotary encoder to control its duty cycle, implement a program letting you control the brightness of an LED by rotating the rotary encoder. Do not use `pinMode()` or `analogWrite()`.

➔ DEMO: E2_T2.MOV
◆ https://drive.google.com/file/d/1eSqzfP96Rv3JoPsYQaYHjHNzQQtHqUvt/view?usp=drive_link

**Discussion for task 2:**

- Design of the interrupts:
  - Start with the logic flow:
    - 1. CLK or SW pin sends a RISING voltage
    - 2. Encoder or Switch ISR is being executed: increase/decrease a value accordingly, or reset the counter, respectively.
  - For encoder ISR:
    - The encoder starts with IDLE state.
    - The first ISR changes the state to TURNING.
    - Depending on the value of DT pin at the moment of interrupt, we increase/decrease the value accordingly
    - The second ISR changes the state back to IDLE, preventing debouncing or double input (this is not entirely perfect but it works!).
- Challenges:
  - Hard to understand documentation and material. The mathematics makes sense after a while for me, but it is not easy to pick it up in a short time for the lab.
  - Lots of confusion surrounding the register variable name.
- "If you were in a lab with access to more tools how would you test to ensure that your PWM code is working?"
  - I can use an oscilloscope to measure the voltage output overtime of the PWM pin. I expect to see square waves with the correct duty cycle that the code sets it to be.

**Question 2**: Because `Timer2` has only 8 bits, it overflows much more frequently than Timer1. If you were to implement the blink example from Lab 1 using `Timer2`, what is the least frequently that the LED could blink? What would the `OCRx` and prescaler values be? Show your calculations.

➔ **ANSWER**:
◆ The slowest possible LED blink using Timer2 in Overflow Mode occurs with prescaler = 1024, where the timer overflows every:

$$T_{overflow} = 1024 * 256 / 16e6 = 16.38\ (ms)$$

- Counting 256 overflows, the total blink period (ON + OFF) is:

$$T_{blink} = 256 * 16.38 = 4.19\ (s)$$

- Thus, the longest blink cycle is 4.19 seconds.
- NOTE: this answer was partially supported by explanations of Arduino Overflow Mode, AVR Timer, OCR, and CTC from ChatGPT 4.0.

# Exercise 3: Test your Reaction Time

**Task**: Implement the reaction time game according to the specifications given above. Use an interrupt for the button and Arduino timer interrupt for the time limit. (Edit March 4) Read from the joystick in a non-blocking manner.

➔ DEMO: E3.MOV
- https://drive.google.com/file/d/1U4aStLGAat1k7JIFWvsAsA63Z1UWKGGY/view?usp=drive_link

➔ Note: the design of the "game" is with the code comments!

# Exercise 4: Door Safety System, Revisited

**Task**: Starting with your Lab 2 code, refine your design so that it is more efficient. Make use of timers and interrupts.

➔ DEMO: E4.MOV
- https://drive.google.com/file/d/1D80fnmUVVtiW0m7i_kGLo9toyWt8demh/view?usp=drive_link

Note: due to my previous implementation being mainly composed of functions from different libraries, the only `interrupt` I could implement was the tamper sensor - the magnetic sensor. The video discusses further into this, plus the potential usage of timers as I didn't have enough time to finish the lab.

⭐ Have a happy day!