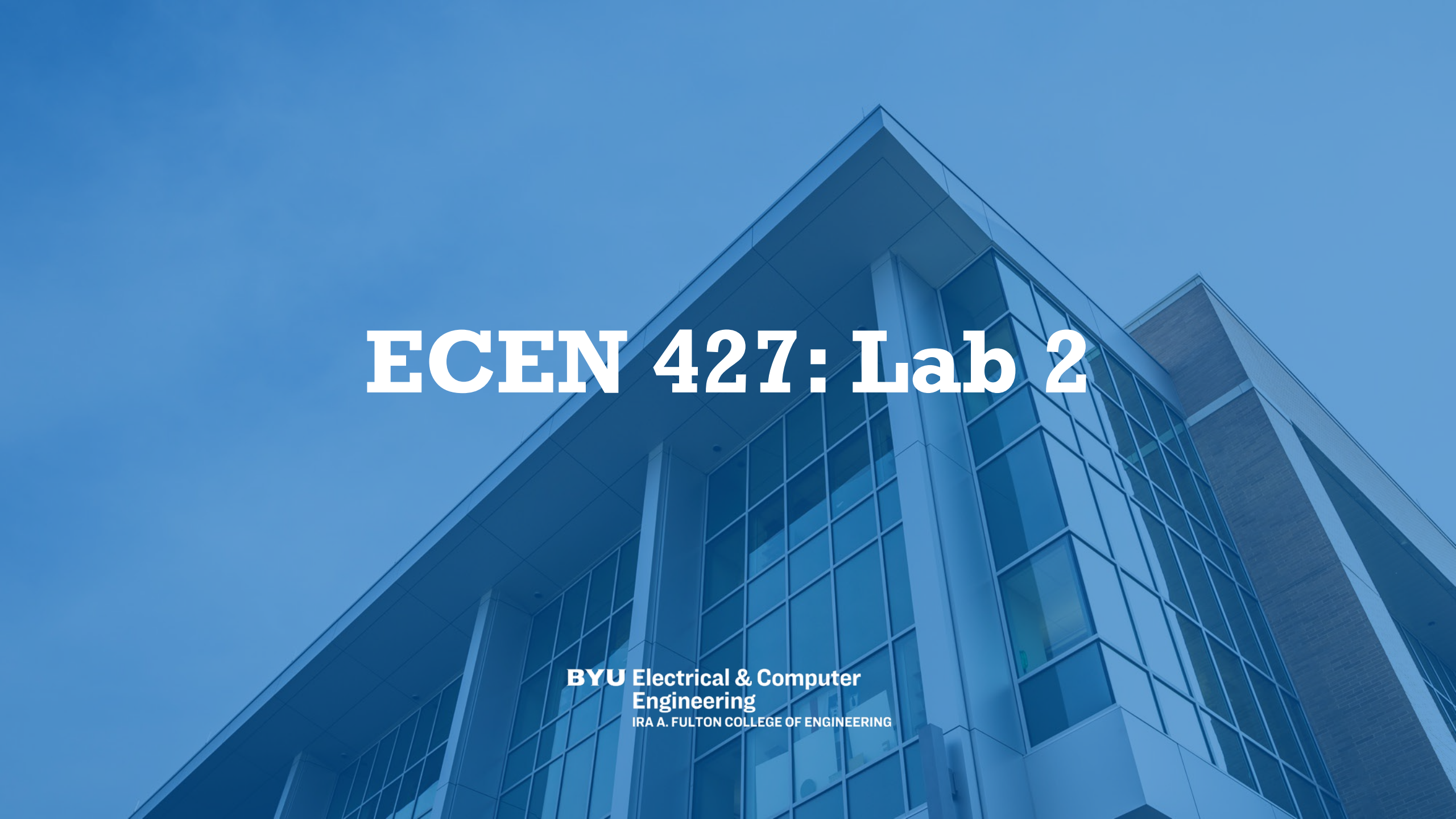


ECEN 427: Lab 2

BYU Electrical & Computer Engineering
IRA A. FULTON COLLEGE OF ENGINEERING

What's a driver?

In Lab2, you must create drivers for:

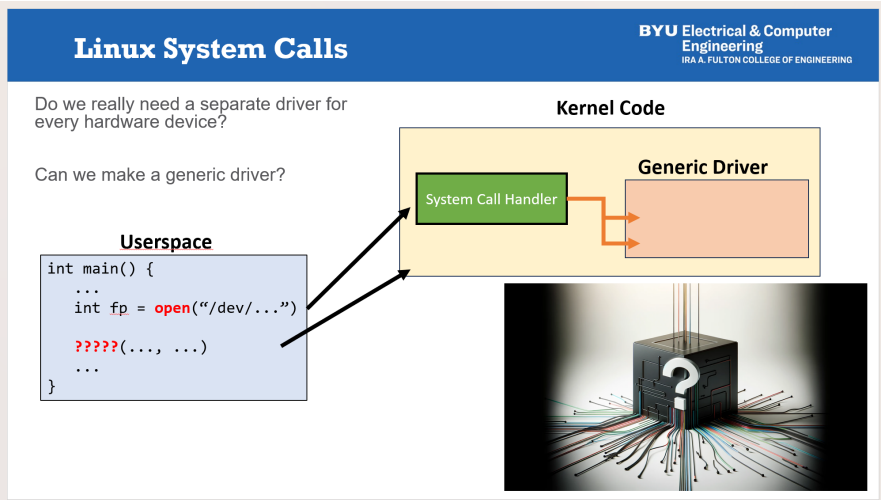
1. Buttons
2. Switches (same as buttons)
3. Interrupt Controller

https://github.com/byu-cpe/ecen427_student/tree/main/userspace/drivers

Review

What is the User space I/O (UIO) driver in Linux?

- Provides generic user space access to a hardware device
- Allows drivers to be written
- Knows nothing about the hardware device, except its physical base address and interrupt line



What API (functions) does it provide to user space?

- Read/Write device addresses (via mmap)
- Enable interrupt line on CPU (via write)
- Wait/Check for interrupt activation (via read/poll)

Hardware System

- https://byu-cpe.github.io/ecen427/documentation/hardware/

GPIO

1. How do you read/write to the GPIO controller registers?

    1. Open the appropriate device file (see system.h) → fd
    2. Use mmap: char * ptr = mmap(…, fd, …) → ptr
    3. Use pointer to access registers (keeping in mind rules of pointer arithmetic)

2. How do you use the GPIO controller registers to read the current value of the buttons?

    1. Read from the data register (base address 0)

    uint8_t buttons = *ptr;

GPIO

1. How do you enable the GPIO controller to generate an interrupt when a button is pressed?
    1. Set bit in global interrupt enable register
    2. Set bit in per-channel interrupt enable register

2. For how long will the GPIO controller generate an interrupt?

    It is a “level” interrupt, so it stays high until acknowledged

3. How do you acknowledge the interrupt?

    Write a ‘1’ to the status bit, which is toggle on write (TOW). This will cause it to flip from 1 to 0.

4. Is an interrupt generated on press? On release? Or both?

    Both. Any change in *any* of the buttons will generate an interrupt. If an interrupt is already active, it has no effect.

Interrupt Controller

1. What is an interrupt controller?

   Aggregates multiple interrupt lines from different hardware devices,
   and combines them into a single interrupt output

2. How many interrupt lines does the user space interrupt controller watch?

   Aggregates 4 interrupt lines (FIT, Buttons GPIO, Switches GPIO, DMA)

3. Suppose you want the interrupt controller to generate an interrupt when an interrupt is detected on input 1. What do you need to do to set this up?

   1. Set bits in the MER
   2. Enable bit 1 of IER (read IER, set bit 1, write back OR … write to SIE)
   3. Using write() system call on the UIO device file for the intc.

Interrupt Controller

1. For how long will the interrupt controller generate an interrupt?

   Generate an interrupt forever, until it is acknowledged.

2. What do you need to do to stop the interrupt controller from sending an interrupt?

   Acknowledge the interrupt:
   - Write an appropriate bit to the IAR

3. If a button is pressed in our system (and assuming interrupts are enabled), what do you need to do to handle it?

   Acknowledge the interrupt:
   - Acknowledge the interrupt from the GPIO → buttons_ack_interrupt()
   - Acknowledge the interrupt from the INTC → intc_ack_interrupt()
   - Re-enable input to the CPU (via write syscall) → intc_enable_uio_interrupts()