Embedded Linux Device Driver

Interrupt handling

- Top halfs and Bottom halfs -- Refer slides and demos

Kernel Threads

- Like user space we can create threads in kernel space as well.
- The kernel threads should be used only for "dedicated" task/sub-system.
- Kernel threads use more resources and hence not advised to use often.
- To create a new kernel thread:

```
struct task_struct *kthread_create(int (*threadfn)(void *data),
                                   void *data,
                                   const char namefmt[],
                                   ...);
```

- However, this thread needs to start explicitly using wake_up_process() call.
- Alternatively, a kernel thread can be created and started in a single call,

```
struct task_struct *kthread_run(int (*threadfn)(void *data),
                                   void *data,
                                   const char namefmt[],
                                   ...);
```

- Example Kernel Thread:

```c
// thread function
static int led_blink_thread(void *data) {
    int i, state = 1;
    for(i=1; i<=20; i++) {
        gpio_set_value(LED_GPIO, state);
        state = !state;
        msleep(500);
    }
    return 0;
}
```

```c
// thread creation -- module_init() or ioctl() or ...
struct task_struct *task = kthread_run(led_blink_thread, NULL, "blink_led%d", 0);
if(IS_ERR(task)) {
    printk(KERN_ERR "failed to create led blink thread.\n");
    // ...
}
```

- Kernel threads are managed by "kthreadd" daemon.

Assignment

1. Blink LED when switch is pressed. Use work queues.