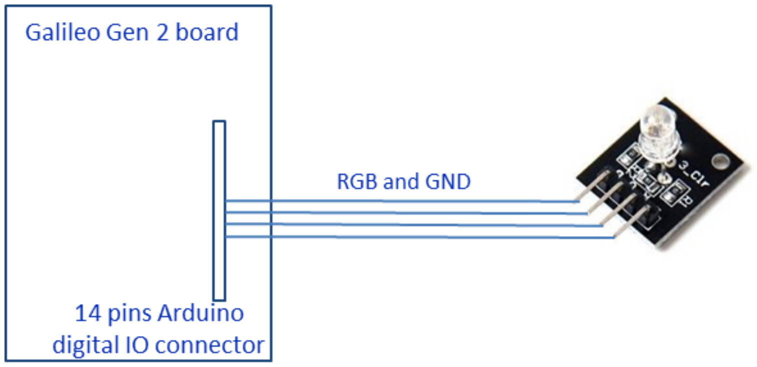
Assignment 3. A GPIO-based LED Driver in Linux (100 points)

Assignment Objectives

1. To learn the basic programming technique for GPIO pins in Linux kernel.
2. To learn timer operations in kernel modules.
3. To learn basic approach to develop char device driver.
4. To learn pin multiplexing mechanisms and programming approaches in embedded systems.

Project Assignment



In this assignment, we will continue the simple LED control of the assignment 2 on Galileo board. Rather than user-level programs, your team will develop a char device driver to control RGB led display. When the driver is installed, a char device "RGBLed" should be created in /dev. The device driver should implement the following file operations on /dev/RGBLed:

- open: to open the RGBLed device.
- write: to write an integer to enable RGB led display. The 3 least-significant bits indicated whether the R, G, and B leds are ON or OFF.
- ioctl: to implement the "CONFIG" command for configuring the RGB led. The argument consists of 4 integers to specify the intensity and the pin connection of the R, G, and B leds. For instance, the input "50, 0, 1, 2" indicates that the LEDs will be on for 50% and off for 50% in every cycle duration, and the LED pins are connected to IO0, IO1, and IO2 of the digital connector. If any of the parameters have invalid values, -1 is returned and errno is set to EINVAL. Also, we will consider to connect a LED to any 3 of the 14 digital pins.
- release: to close the opened RGBLed device.

To control LED intensity for each color, you will need to put out either 0 or 1 on the corresponding GPIO pins in each cycle intervals. For instance, for 50% intensity, the GPIO pin to a LED is changed to 0 after it has been set to 1 for a duration of half cycle. For the assignment, you can set the cycle duration of intensity control to 20ms. As a kernel module, you can initiate a timer with a proper callback function to alter LED signals. The kernel timer in /kernel/time/timer.c, or the hrtimer in /kernel/time/hrtimer.c can be good candidates to use for this purpose. The programming examples can be found in

https://www.cs.bham.ac.uk/~exr/lectures/opsys/12_13/examples/kernelProgramming/kernel/myTimer.c

https://gist.github.com/itrobotics/596443150c01ff54658e

To control gpio pins in the kernel, you cannot use the same sysfs interface as you did in Assignment 2. However, the high-level gpio driver "gpiolib.c" provides an API to access the corresponding gpio hardware drivers and to control apio operations. A good reference can be found at https://lwn.net/Articles/532714/.

To test the driver, you will need to write a user-space application to display a lighting sequence of {OFF, R, G, B, R&G, R&B, G&B, R&G&B} where each step of the lighting sequence is with a duration 0.5 second. The lighting sequence should be displayed repetitively and can be interrupted by any mouse button click. If there is a button click, the display should immediately move to the beginning of the sequence.

Since the RGB pins can be connected to any 3 IO pins of the digital connector, your LED driver must configure the pin multiplexing properly. It is likely that a pin multiplexing table is needed in which you can define how the IO pins must be set for input/output mode, and how the GPIO function is selected.

Due Date

The due date is at 11:59pm, Oct. 23.

What to Turn in for Grading

- Create a working directory, named "ESP-teamX-assgn03", for the assignment to include your source files, makefile(s), and readme. Compress the directory into a zip archive file named ESP-teamX-assgn03.zip. Note that any object code or temporary build files should not be included in the submission. Submit the zip archive to Blackboard by the due date and time.
- Please make sure that you comment the source files properly and the readme file includes a description about how to make and use your software. Don't forget to add each team member's name and ASU id in the readme file.
- There will be 20 points penalty per day if the submission is late. Note that submissions are time stamped by Blackboard. If you have multiple submissions, only the newest one will be graded. If needed, you can send an email to the instructor and TA to drop a submission.
- Your team must work on the assignment without any help from other teams and is responsible to the submission in Blackboard. No collaboration between teams is allowed, except the open discussion in the forum on Blackboard.
- Failure to follow these instructions may cause deduction of points.
- Here are few general rule for deductions:
  - No make file or compilation error -- 0 point for the part of the assignment.
  - Must have "–Wall" flag for compilation -- 5-point deduction for each warning.
  - 10-point deduction if there is no instruction on compilation or execution in README file.
  - Source programs are not commented properly -- 10-20-point deduction.
- ASU Academic Integrity Policy (http://provost.asu.edu/academicintegrity), and FSE Honor Code (http://engineering.asu.edu/integrity) are strictly enforced and followed.