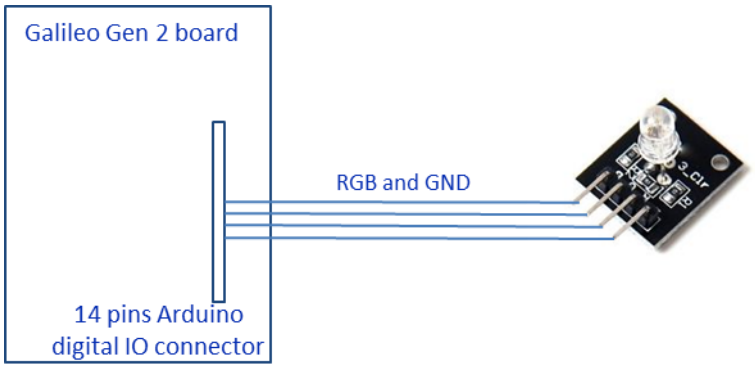
Assignment 2. GPIO control in Linux (200 points and 50 bonus points)

Assignment Objectives

1. To learn the basic programming technique for GPIO pins in Linux system.
2. To learn IO control in bash script, user-level programs, and kernel modules.
3. To learn GPIO control via the interface of GPIO core in Linux kernel.
4. To learn pin multiplexing mechanisms and programming approaches in embedded systems.

Project Assignment

In Linux systems, you can development various kinds of programs to perform IO operations. In this assignment, you are required to write a bash script program, a user-space program, and a kernel module to interface with a RGB LED. The programs can control LED luminous Intensity, as well as the combination of red, green and blue colors. As shown in the following diagram, the LED's red, green, and blue pins are connected to any of 3 GPIO pins of the Arduino digital IO connector of Galileo Gen 2 board.



The input arguments to your programs consist of 4 integers to specify the percentage of duty cycle of a PWM signal for luminous Intensity, and the digital IO pins that the RGB LED pins are connected to. 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. For the assignment, you can set the cycle duration of intensity control to 20ms. Also, if any of the input integers is invalid, your programs should print out an error message and terminate.

The RGB LED control that your programs must demonstrate is to apply the PWM signal to the combinations of one, two or all three LEDs in a 7-step periodic lighting sequence, where each step is with a duration of 0.5 second. At the end of each step, your programs must check any mouse button click. If there is no button click, the sequence should continue. Otherwise, the program execution should be terminated. An example lighting sequence could be {R, G, B, R&G, R&B, G&B, R&G&B}.

Since the RGB pins can be connected to any 3 IO pins of the digital connector, your programs 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.

Part 1 – A user-space application for RGB LED display

In this part, your team should develop a user-space application program to perform the aforementioned RGB display operations. The main program is named as *RGBLed.c*. If needed, additional source programs and header files can be included in your development.

To control LED intensity and the display duration for each color, you will need to put out either 0 or 1 on the corresponding GPIO pins for certain intervals. For instance, for 50% intensity, the GPIO pin to red LED is changed to 0 10ms after it is set to 1. To perform such timed actions, you can consider to use usleep or nanosleep function, or a POSIX timer. In addition, please make sure to use macros for any literal constants or flags, e.g., *"#define cycle_duration 20*".

Part 2 –A kernel device driver for LED display and intensity control

In this part, you 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. If any of the parameters have invalid values, -1 is returned and errno is set to EINVAL.
- *release:* to close the opened RGBLed device.

To control the intensity, it is needed to turn LED signals on and off periodically. 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 test the driver, you will need to write a user-space application to display the lighting sequence of {R, G, B, R&G, R&B, G&B, R&G&B} repeatedly until a mouse button click and have each step of the lighting sequence is with a duration 0.5 second. The input arguments to your program are 4 integers for display intensity and the three IO pins that the R, G, and B leds are connector to.

Extra Task – A Bash shell script for RGB LED display (50 bonus points)

In this extra task, you will develop a script RGBLed.sh to perform the display function stated above. A Bash shell script is a program written in the Bash programming language and the commands in a script are executed sequentially. To implement the required display function, you may need to exercise few features of Bash script, including looping, array, IO, timeout, sleep, and function call. The following document may be helpful to get started:

- http://www.tldp.org/HOWTO/pdf/Bash-Prog-Intro-HOWTO.pdf
- http://www.tldp.org/LDP/abs/abs-guide.pdf

Due Date

The due date for Part 1 is 6pm, Oct. 12. For Part 2, the due date is 6pm, Oct. 21.

What to Turn in for Grading

- Create a working directory to include your source files (.c and .h), makefile(s), readme file. Compress the directory into a zip archive file named cse438-teamX-assgn02-partN.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.
- If you choose to do the extra task, your implementation, as a script file, should be included in the submission of Part 2.
- 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 no compilation or execution instruction 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. A grade XE will be assigned to any cases of AIP violation.