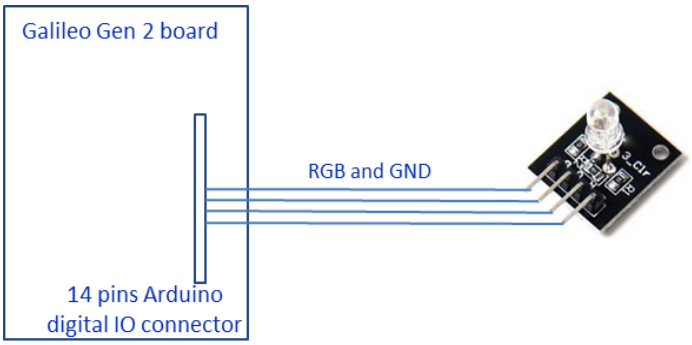
Assignment 2. GPIO control in Linux (100 points)

Assignment Objectives

1. To learn the basic programming technique for GPIO and PWM pins in Linux system.
2. 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 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.



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 RGB display operations described below. 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. 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 gpio signals to the combinations of one, two or all three LEDs in a 8-step periodic lighting sequence {OFF, R, G, B, R&G, R&B, G&B, R&G&B}, where each step is with a duration of 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 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.

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 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. Thus, each step will consist of 25 cycles.

The main program is named as *RGBLed_1.c*. If needed, additional source programs and header files can be included in your development.

Part 2 – PWM Signals for LED display and intensity control

In this part, you will modify the program in part 1 to perform the same RGB led display except:

- Instead of gpio signals, you should use PWN signals to control the LEDs.
- Instead of skipping to the beginning of sequence, mouse clicks are used to control LED intensity. Each click of left/right button increases/decreases duty cycle by 10%.

The main program for part 2 is named as *RGBLed_2.c*. The input arguments to your program are still 4 integers for the initial display intensity and the three IO pins that the R, G, and B LEDs are connector to. Also, if any of the input integers is invalid, your programs should print out an error message and terminate.

Questions:

1. After you write 74 to /sys/class/gpio/export, will you receive an error when opening /sys/class/gpio/gpio74/direction? Explain your answer.
2. Can we receive an interrupt when there is a rising edge on gpio38 which is connected to IO7 directly on Galileo Gen2 board? Explain your answer.
3. After you export 3 to /sys/class/pwm/pwmchip0/export, what is the period of pwm3 signal? Can you modify it? Explain your answer.

Due Date

The due date is at 11:59pm, Oct. 1.

What to Turn in for Grading

- Create a working directory, named "ESP-teamX-assgn02", for the assignment to include your source files for parts 1 and 2, makefile(s), readme, and your answer to the questions (in pdf format). Compress the directory into a zip archive file named ESP-teamX-assgn02.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.