ELEC 424 - Assignment 2: libgpiod

100 points

Read the whole document, there is more than one page! :)

Overview

You will interface your Raspberry Pi with an external circuit implemented on a breadboard via the Pi's GPIO pins. Your software will run in user space by using the libgpiod library in a typical .c file (a user application).

Choose two GPIO pins and set one up to toggle an LED and the other to read from a button/switch. Be sure to use the 330 ohm resistor in series with the LED. Check out the pinouts online or in the slides for the Raspberry Pi to know where to connect wires. Make it so that when the button is open initially, the LED will be on for (approximately) 0.5 seconds and off 0.5 seconds and continue to alternate until the end of time.

Then, if the "button" is pressed and released quickly, that should double the frequency of the LED's alternation between on and off. Any additional quick presses and releases should continue to double the frequency. A long press should result in a return to the original frequency of alternating between on and off. Then it should be able to be accelerated again if the button is pressed quickly. A button press must be detected pretty immediately and cause the LED to turn off while the button is being pressed.

In Ryon B12 (door code: 3090#) [you should have 24/7 card access to Ryon], there will be a small box immediately to your right containing LEDs, 330 Ohm resistors, jumper wires, breadboards, and buttons if you did not get them in class. There is a red sheet of paper in front of this box saying "424 Assignment 2". You can of course use your own supplies. Please feel free to use the same circuit between you and classmates, no need for each of you to have your own breadboard/LEDs/etc. Let me know if anything runs out. Don't return your borrowed supplies to me until the end of the semester, as we will use them for future work.

Note: Be sure to consider button debouncing: One seeming press of the button will generally result in multiple rapid switches between 0 and 1 for the GPIO pin value until it stabilizes, which means that you may not get a clean 0 to 1 or 1 to 0 transition when pressing or releasing. Your code should be robust to this issue, and accuracy in terms of time keeping can be sacrificed as such. In general the code does not need to be exact for timing for this assignment.

Rubric

- **(50 points) In person or submitted recorded video demonstration of functionality. In person demonstrations can be done during instructor or TA office hours.**
    - (10 points) LED toggles on and off
    - (10 points) Initial LED toggling is approximately 0.5 s on and 0.5 s off
    - (10 points) Quick (you can be the judge of quick) button press causes LED toggling frequency to double
    - (10 points) Long (again, you can be the judge of long) button press causes LED toggling pattern to return to 0.5 seconds on and 0.5 seconds off
    - (10 points) Further quick button presses continue to double toggling frequency as expected
    - Recorded video demonstration submission
        - If you prefer to submit a video demo, please upload it to the Box folder that I shared with your Rice email. It is your responsibility to let me know 48 hours before the deadline if you do not have access to the folder, otherwise it is not my responsibility if you have to submit late. Video files submitted must be named as the following (replacing words as appropriate): ELEC424_Assignment2_Firstname_Lastname_netID
            - Note: I have set the privacy to where I believe no one can view each others' videos, however you will be able to see who has uploaded and who has not. If this is a concern for you, please meet with us to do an in person demonstration instead.
- **(50 points) Submission of relevant commented code files and report to Canvas**
    - (10 points) Code attempts to achieve objectives/requirements stated in instructions
    - (10 points) Code reasonably includes comments (at least every other line includes a comment)
    - (5 points) The following file(s) must be submitted in source form (.tbl, .c, etc.) - not a PDF
        - Your main .c file
    - (25 points) PDF report that includes:
        - (1 point) Title of assignment/project
        - (1 point) Your name
        - (5 points) 1 paragraph (at least 4 sentences) describing the goal of the project and the steps you took to complete it (include a statement on each key function)
        - (5 points) A 2nd paragraph (at least 4 sentences) describing what you found challenging/any bugs you had to fix, what you learned, and what you think would be another interesting application for GPIO.

- (5 points) After completing the coding and demonstration parts of the assignment, I grant permission for you to ask chatGPT to complete an assignment similar to this one. Ask chatGPT how to make a gpio C program for the Raspberry Pi Zero W using libgpiod (no need to give it all the details, you can ask for a simple version). Write a 3rd paragraph (at least 4 sentences) describing what chatGPT tells you, how accurate you think it is, and how it seems to differ from your solution.
    - You must include a shared link to your chatGPT chat (instructions) and cite chatGPT according to the standard of this document.
- (4 points) Include a screenshot showing a significant portion or all of your code.
- (2 points) Include a screenshot of terminal output showing the messages printed by your code.
- (2 points) All screenshots in the report must include a figure label with a short description.

Guidelines

- Install the necessary package on your Pi: `sudo apt install libgpiod-dev`
- Your code must print out a message to the console every time that a button press is detected, and state which type (short or long) is detected
- Consider my incomplete code below for some of the most relevant functions in libgpiod:

```
// Declare variables
const char *chip_title = "gpiochip0";
struct gpiod_chip *chip;
struct gpiod_line *line_output;
struct gpiod_line *line_input;

// Open chip that handles GPIO
chip = gpiod_chip_open_by_name(chip_title);

// Grab lines from chip for output (LED) and input (button)
// v and w should be GPIO numbers - 5 and 6 are my suggestions
line_output = gpiod_chip_get_line(chip, v);
line_input = gpiod_chip_get_line(chip, w);

// Specify which line is output and which is input
// The “something” is a name that doesn’t matter for us
// z is the initial output value, can be 0 or 1
gpiod_line_request_output(line_output, "something", z);
gpiod_line_request_input(line_input, "something");

// The below functions set values
```

    - `gpiod_line_set_value(line_output, t)`
    - `gpiod_line_set_value(line_output, y)`

    - **`// The below function reads input`**
    - `button_state = gpiod_line_get_value(line_input);`

    - **`// The following two lines close things up for you`**
    - `gpiod_line_release(line);`
    - `gpiod_chip_close(chip);`

- **You will have to include the flag -lgpiod at the end of the compile command when running gcc, otherwise it will say gpiod functions are undefined**
- You can get inspiration from online examples, but do not directly copy their code
- You may talk with classmates about the assignment, but do not share or copy code
- Although libpgiod is currently hosted on kernel.org, you can easily view the relevant (albeit older) gpio.h header file on GitHub here to know more about what the input/output arguments are for the above functions
- Feel free to use any typical libraries for timing