

FALL 2023
MOBILE AND EMBEDDED SYSTEM DESIGN AND APPLICATION
(COMP/ELEC 424/553)

Assignment 2: libgpod

Shaun Lin (hl116), S01435165

The goal of this project was to use the gpod library and write a C script in Raspberry Pi to control LED blinky frequency with a button. I used GPIO pin 5 for LED and GPIO pin 6 for Button, the LED was initialized on/off every 0.5 seconds. If you quickly press the button less than 1 second, it will double the blinky rate, if you press the button more than 1 second will reset the blinky frequency to every 0.5 seconds on/off. I custom wrote a function to measure the button press duration, this allows my code to be clean and easier to understand.

My challenge was trying to turn on the LED, initially I used GPIO pin 17 for LED, but it's not working with the gpod library, after Dr. Joseph Young suggested that I use GPIO pin 5 and 6, I successfully turned on the LED and got an interrupt signal from the button. Through this project, I learned how to interface hardware using Linux GPIO and using the gpod library to easily control external peripherals.

There's an interesting application could make this assignment more interesting, we can write a PWM signal to LED, make LED doing a "breathing light", this would allow us to understand how to control GPIO with PWM signal, then we can also control the motor using PWM signal and build a RC car.

I googled "Raspberry Pi gpod library LED blinky example", and found [this website](#), it's basically the same as instruction's guidelines. However, I asked chatGPT to give some suggestions about how to implement the logic to make the LED initially blinky with 0.5 seconds frequency and pressed the button to double speed and pressed long to reset the blinky frequency. Overall the ChatGPT code seems like it would work. However, after I tested it's not working good, it handles doubling the blink rate and resetting in the main loop, I don't like this style, so I write a `measure_press_duration()` function outside the `main()` function to check the duration and handle doubling. Furthermore, ChatGPT used two while loops to separate the repetitive LED blinking and used `gpod_line_event_wait()` to wait for button presses in the inner loop, this would make the control flow more complex and hard to understand.

In conclusion, ChatGPT provided the sample code which does not work well. However, my own implementation required more robust logic and easier to understand. I totally took about 4 hours on this project, it's very interesting and I learned a lot about how to interfacing peripherals using GPIO.

Acknowledgements

I acknowledge the use of ChatGPT-3.5 to generate a few initial scripts for this project.

I entered the following prompt: *“Hi ChatGPT, I'm working on a raspberry pi projcet, write a C language script that when the "button" is open initially, the LED will be blinky for 0.5 seconds.*

Then, if the “button” is pressed less than 1 second, that should double the blinky rate. Any additional presses less than 1 second should continue to double the blinky rate until I pressed the button over 1 second, then LED should reset back to original blinky rate 0.5 seconds.

How should I modify my following code that used libgpod library to complete above task?” I used the output to help me quickly understand how to detect the button state, and use portions of the output as a starting script. I modified the output generated, discarding several suggestions and replacing them with my own ideas based on the research and reading I completed.

Additionally, I referenced [this source](#) to test the LED blinky program, and based on the code made a few changes and added to the above prompt.

Finally, I prompted ChatGPT-3.5 to optimize my original text and made small changes based on its suggestions.

Prompt: "Hi ChatGPT, I'm working on a raspberry pi projcet, write a C language script that when the "button" is open initially, the LED will be blinky for 0.5 seconds.

Then, if the “button” is pressed less than 1 second, that should double the blinky rate. Any additional presses less than 1 second should continue to double the blinky rate until I pressed the button over 1 second, then LED should reset back to original blinky rate 0.5 seconds.

How should I modify my following code that used libgpod library to complete above task?".

Open AI, ChatGPT-3.5, October 22, 2023.

Appendix A

(4 points) Include a screenshot showing a significant portion or all of your code.

```
83
84 // Function to measure button press duration
85 unsigned long measure_press_duration(struct gpiod_line *button)
86 {
87     unsigned long duration = 0;
88     const struct timespec delay = {0, 100}; // 100 nanoseconds
89
90     // Wait for button to be pressed if not already pressed
91     while (gpiod_line_get_value(button) == 1)
92     {
93         nanosleep(&delay, NULL);
94     }
95
96     // Measure the duration of the button press
97     while (gpiod_line_get_value(button) == 0)
98     {
99         nanosleep(&delay, NULL);
100         duration += 100; // Add 100 nanoseconds to duration
101     }
102
103     return duration;
104 }
```

Figure 1. Screenshot measure_press_duration() function of main.c

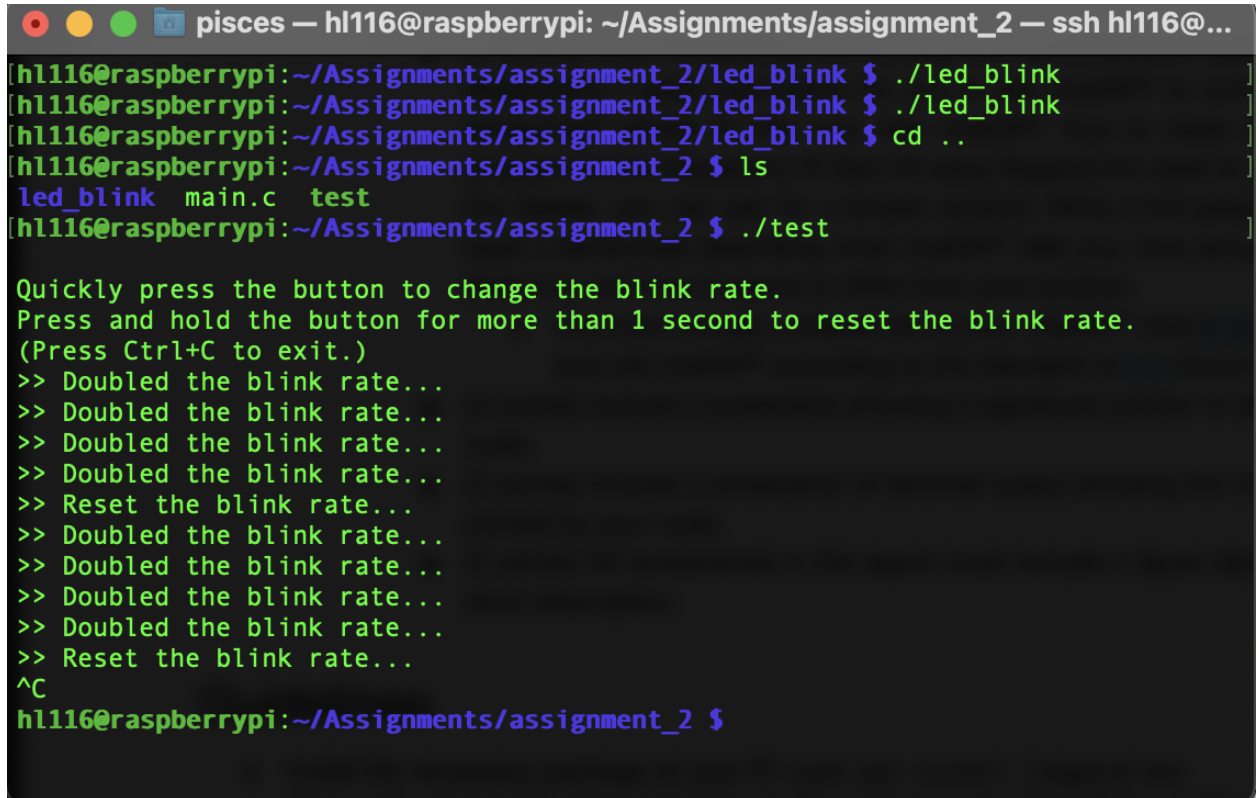
```

C main.c > main()
1 //////////////////////////////////////////////////
2 // COMP/ELEC 424/553
3 // Assignment 2
4 // Authors: Shaun Lin(hl116)
5 //////////////////////////////////////////////////
6
7 #include <gpio.h>
8 #include <stdio.h>
9 #include <unistd.h>
10 #include <time.h>
11
12 unsigned long measure_press_duration(struct gpio_line *button);
13
14 int main()
15 {
16     // Define the GPIO chip and line variables
17     const char *chip_title = "gpiochip0";
18     struct gpio_chip *chip;
19     struct gpio_line *led;
20     struct gpio_line *button;
21
22     // Define the variables for the LED state, counter and time
23     int state, counter;
24     unsigned long blink_rate = 500000; // in microseconds (0.5 s)
25
26     // Setup the GPIO Chip struct...
27     chip = gpio_chip_open_by_name(chip_title);
28     // Setup the GPIO "line" (pin 5)
29     led = gpio_chip_get_line(chip, 5);
30     // Setup the GPIO "line" (pin 6)
31     button = gpio_chip_get_line(chip, 6);
32     // Set the LED "line" for output
33     gpio_line_request_output(led, "GPIO 5 Output", 0);
34     // Set the button "line" for input
35     gpio_line_request_input(button, "GPIO 6 Input");
36
37     // Initialize the LED state
38     state = 0;
39
40     // instructions
41     printf("\nQuickly press the button to change the blink rate.\n");
42     printf("Press and hold the button for more than 1 second to reset the blink rate.\n");
43     printf("(Press Ctrl+C to exit.)\n");
44
45     while (1)
46     {
47         // Button is not pressed
48         if (gpio_line_get_value(button) == 1)
49         {
50             gpio_line_set_value(led, state);
51             usleep(blink_rate);
52             state = !state;
53         }
54         // Button is pressed
55         else
56         {
57             // Measure the duration of the button press
58             unsigned long press_duration = measure_press_duration(button);
59             // Debugging
60             // printf("Button is pressed for %lu microseconds\n", press_duration);
61
62             // If button is pressed for less than 1 s
63             if (press_duration < 1000000)
64             {
65                 blink_rate /= 2; // Double the blink rate
66                 printf(">>> Doubled the blink rate...\n");
67             }
68             else
69             {
70                 blink_rate = 500000; // Reset to original rate
71                 printf(">>> Reset the blink rate...\n");
72             }
73             usleep(20000); // Sleep for 20 ms to prevent bouncing
74         }
75     }
76
77     // Release lines and chip & exit
78     gpio_line_release(led);
79     gpio_line_release(button);
80     gpio_chip_close(chip);
81     return 0;
82 }

```

Figure 2. Screenshot main() function of main.c

(2 points) Include a screenshot of terminal output showing the messages printed by your code.



```
pisces — hl116@raspberrypi: ~/Assignments/assignment_2 — ssh hl116@...
[hl116@raspberrypi:~/Assignments/assignment_2/led_blink $ ./led_blink ]
[hl116@raspberrypi:~/Assignments/assignment_2/led_blink $ ./led_blink ]
[hl116@raspberrypi:~/Assignments/assignment_2/led_blink $ cd .. ]
[hl116@raspberrypi:~/Assignments/assignment_2 $ ls ]
led_blink  main.c  test
[hl116@raspberrypi:~/Assignments/assignment_2 $ ./test ]

Quickly press the button to change the blink rate.
Press and hold the button for more than 1 second to reset the blink rate.
(Press Ctrl+C to exit.)
>> Doubled the blink rate...
>> Doubled the blink rate...
>> Doubled the blink rate...
>> Doubled the blink rate...
>> Reset the blink rate...
>> Doubled the blink rate...
>> Doubled the blink rate...
>> Doubled the blink rate...
>> Doubled the blink rate...
>> Reset the blink rate...
^C
hl116@raspberrypi:~/Assignments/assignment_2 $
```

Figure 3. Screenshot of terminal output