Week 7 Lab

CC2511

Task Description

You are to develop a command-line user interface for the week 5 task (controlling the LED brightness). You must extend your code to perform PWM across all three colour channels, and then provide a textual user interface that allows the user to:

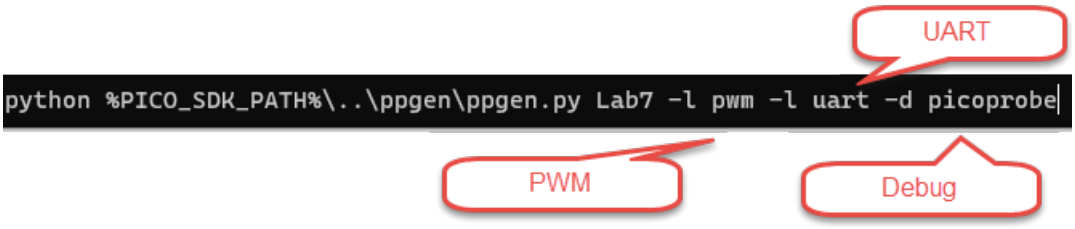
- See the current PWM ratios, and
- Type in new PWM ratios for each channel using commands like "`red 10`" (followed by the Enter key). This command would set the red PWM ratio to `10/255`.

An example implementation is shown below:

NOTE—You do not need to reproduce this visual style. A minimal implementation (suitable to pass the lab) needs only to show the three PWM ratios and allow text commands to be entered to control these.

Project

- When you create the project, you can use `-l pwm` to add PWM support and `-l uart` to add UART support.
- This program can be reasonably complex. The ability to debug it is valuable. Add the parameters `-d picoprobe` when you create the project:



You will be able to deploy and run the program from VS Code simply by pressing F5: there is no need to reboot the board or to copy the uf2 file.

Hints

- Refer to the Week 7 Lecture notes for advice on finding and using suitable example code.
- Set up and enable an interrupt handler for UART input.
- In the event handler, populate the input buffer.
- Do not block during the interrupt handler:- the ISR may lock up the program and never return.
  - Writing to the UART can block the program.
  - Observe how the example code uses `uart_is_writable()` to avoid this.
- Your main program loop should use the "wait for interrupt" assembly code instruction to enter a low power sleep state.

```
__asm ("wfi");
```

- You should accumulate received characters into a string buffer. Once you detect the enter key has been pressed, add a NULL termination to your buffer and then interpret it with the **`sscanf`** and/or **`strcmp`** functions. Use **`sscanf`** if the command has arguments, and **`strcmp`** if the command is always the same static text.
- Declare three unsigned 8-bit variables to hold the current red, green and blue brightness values. These can contain values from 0 to 255 inclusive. e.g.:

```
uint8_t red, green, blue;
```

- As in previous labs set the actual PWM level to the square of the value, to produce a better distribution, e.g.:

```
pwm_set_gpio_level(RED_PIN, red * red);
```

- To read a `uint8_t` data type with `sscanf`, use the `%hhu` modifier. If you use the wrong format specifier then `sscanf` will overwrite other adjacent variables and cause unpredictable behaviour.
- The `terminal.h` header file on LearnJCU provides ready-made functions for setting the cursor coordinates and changing the text and bcakground colours. The example shown above used the `term_setcolor()` and `term_move_to()` functions in `terminal.h` to draw the coloured boxes.

for (;;) {
if (UART received char) { ... }
if (switch pressed) { ... }
/* ... many others ... */
}

NOTE—You can open multiple VS Code sessions for multiple projects and copy/paste code between projects.

+ buffer index

interrupt service routine

buf[buf_index] = /* new char from UART */;
buf_index++;

WHERE ?

already done

nah fam

what for?

Assessment

To finish this lab, demonstrate the following to your tutor:

- A working command-line user interface where arbitrary PWM ratios can be typed in for each colour channel.
- The current value of the three PWM channels displayed on the screen.
- Show your prac tutor your GitHub webpage where your source code is uploaded.

HINT—Your solution to this lab might be useful in your assignment. Solutions to the optional extensions might also be useful.

Optional Extension

- Make it possible to erase typing mistakes with backspace. By default on PuTTY, the backspace key sends `0x7f`.
- Correctly handle the case where the user types in a very long command. You will need to ensure that you do not overflow the string buffer.
- Display a visually pleasing interface like the example screenshot above.
- The sequence Ctrl-L is sometimes used in terminal programs to request that the screen be completely redrawn. This is useful if you have disconnected and reconnected PuTTY for example. Ctrl-L will send the character `'\f'`.

Second optional extension

- Add event handlers for the three push-buttons on the development board.
- Catch the event when the button is pushed: `GPIO_IRQ_EDGE_RISE`.
- Each button should affect a different colour. When the button is pressed the colour should toggle between maximum brightness and off.
- Look in `pico-examples/gpio/hello_gpio-irq` for example code.

HINT— Look at the dev board's schematic to determine the GPIO pin for each button.

HINT— You can use the same event handler for all three buttons, and inspect the handler's `gpio` parameter.