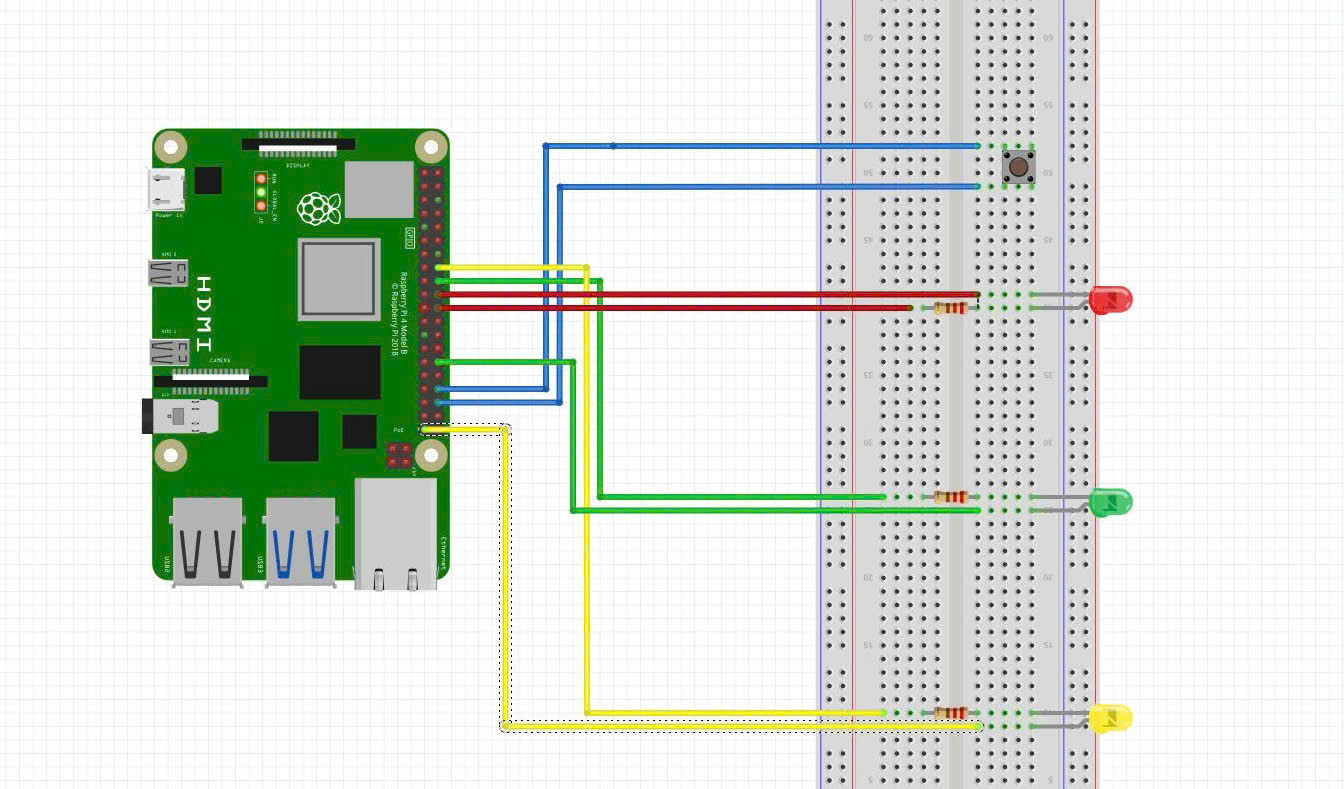
# GPIO LED and Button Driver Project (Mock project)

## 1.Specification

* Multi-LED Control: Independent control of 3 LEDs (Green, White, Yellow)
* Interrupt-driven Button Handling: Advanced button processing with multi-press detection
* Inter-module Communication: Button driver communicates with LED driver
* Dual Interface Support: Both character device and IOCTL interfaces
* Comprehensive User Application: Full-featured test and control application
* Robust Error Handling: Proper cleanup and error recovery mechanisms
* Cross-platform Build System: ARM cross-compilation support



In this project, i add file dtsi directly to the below path instead of overlay method

**/buildroot/output/build/linux-custom/arch/arm/boot/dts/broadcom**

I also have to add the line **#include "gpio\_control.dtsi"** to the file **bcm2711-rpi-4-b.dts.**The detailed dtsi file is added in the assignment folder for declaration.

**Notes: all the pictures in the document are located in the folder pictures of the assigments . Also, the demo video is contained in this folder.**

## 2. Automation setup

Because in this project, the purpose is implement built-in method for automation. In detail, all the platform drivers are compiled and executed without the user interaction like not insmod to kernel space or compiled them. In order to accomplish these feature, i allocate all two drivers in this below path of the buildroot:

**/home/user/Downloads/buildroot/output/build/linux-custom/drivers/misc**

I created a new folder in /misc and add the two driver in it. The architecture is depicted below:

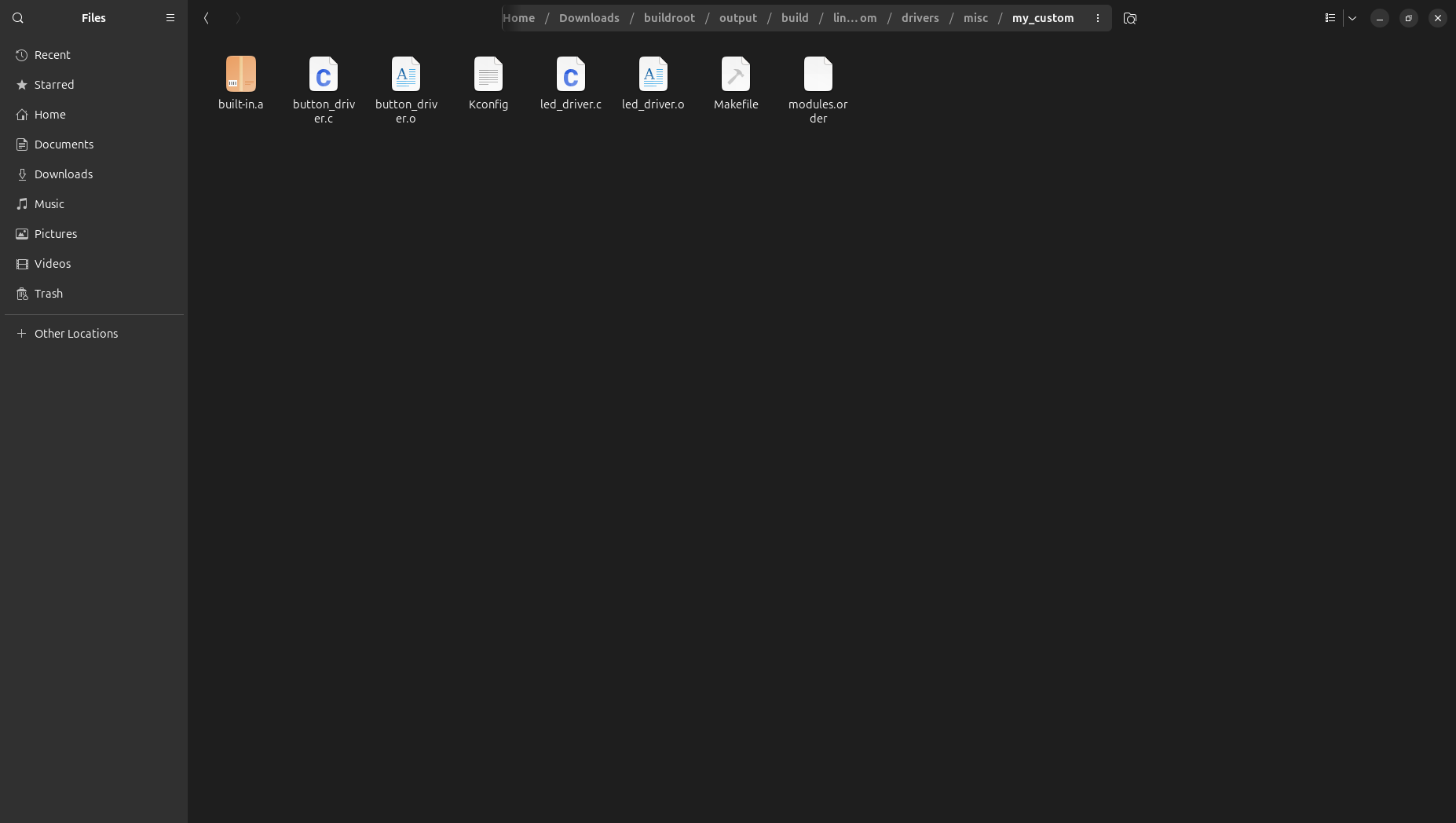
drivers/misc/my\_custom/

├── led\_driver.c

├── button\_driver.c

├── Makefile

├── Kconfig



The content of Kconfig and Makefile in this my\_custom is stored in folder built-in relevant of the assignment folder. Moreover, in the /misc/Kconfig, i add the line in the end of the file

source "drivers/misc/my\_custom/Kconfig"

## 3. Led Driver

* Overview

The LED driver is a platform driver that manages three GPIO-controlled LEDs with comprehensive control interfaces.

* Key Features
* Multiple Device Support: Creates 3 separate device files
* Dual Control Interface: Write operations and IOCTL commands
* State Management: Tracks LED states internally
* Export Functions: Provides GPIO access to other modules ( button driver )
* Proper Resource Management: Uses devm\_\* functions

==> Device file created:

/dev/gpio\_led0 -> Green LED (Minor: 0)

/dev/gpio\_led1 -> White LED (Minor: 1)

/dev/gpio\_led2 -> Yellow LED (Minor: 2)

## 4. Button Driver

* Overview

The button driver implements sophisticated interrupt-driven button handling with multi-press detection and automatic LED control.

* Advanced Features
* Interrupt-driven Processing Edge-triggered GPIO interrupts
* Debouncing: 50ms hardware debouncing
* Multi-press Detection: Counts button presses within timeout window
* Work Queue Integration: Non-atomic processing in work context
* Timer Management: 1000ms timeout for multi-press sequences
* Inter-module Communication: Uses LED driver's exported functions

==> Device file created:

/dev/gpio\_button

* The logic of pressing button:
* 1 press -> LED 0 ON, others OFF
* 2 presses -> LED 1 ON, others OFF
* 3 presses -> LED 2 ON, others OFF
* 4 presses -> All LEDs ON
* 5+ presses -> All LEDs OFF (Reset state)

## 5. User application

* Overview

The gpio\_app application is a command-line tool for controlling 3 LEDs (Green, White, Yellow) and monitoring button status through GPIO on Raspberry Pi. The application communicates directly with kernel drivers through device files.

* The commands of app:

|  |  |  |
| --- | --- | --- |
| **Command** | **Syntax** | **Description** |
| led | ./gpio\_app led <N> <action> | Control specific led |
| all | ./gpio\_app all <action> | Control all led |
| status | ./gpio\_app status | Show overall status |
| button | ./gpio\_app button | Show detailed button info |

**Parameters:**

* <N>: LED number (0, 1, 2)
* <action>: Action (on, off, toggle)

+) Turn on led

./gpio\_app led 0 on

+) Turn off led

./gpio\_app led 0 off

+)Toggle led

./gpio\_app led 0 toggle

+) Turn on all led

./gpio\_app all on

+) Status checking

/gpio\_app status

+) Button status checking

./gpio\_app button

The belowed picture shows the result of the button command

