

# Assignment #3 - Building a Linux Device Driver

Grade Weight: 100pts

Drivers provide a software interface that allows the OS to communicate and control hardware devices. In this assignment, you will learn the fundamentals of building a device driver by implementing a driver that controls the input and output of some physical hardware. More specifically, your driver will control some lights (LEDs) and a button. A user space program will initiate a system call to the driver implemented in the kernel. Based on the input provided by user space process, the driver will turn will control four lights. The driver will also handle hardware signals, but turning off all the lights whenever a physical button is pressed.

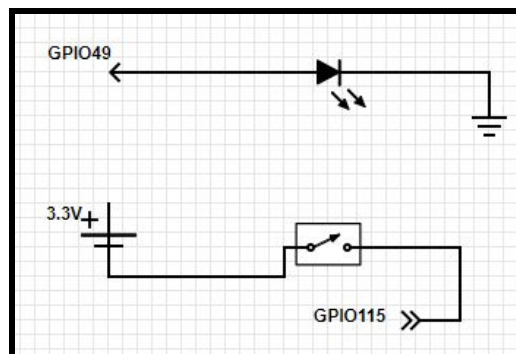
**Hardware Required:** In addition to the Beagle Bone Black, you will also need breadboard to build your circuit. The instructor / TAs will provide you with LEDs, a button and wires to complete this assignment. You are responsible for acquiring your own breadboard, as indicated in the course syllabus.

**Sample Code:** The sample code is a Loadable Kernel Module (LKM) driver. The sample code illustrates a number of critical parts of the assignment. When the button and LED are connected properly to the board, button presses will cause the LED to turn on and off.

- **Sample code is available here:** [Sample Code](#)

When testing the sample code, you will need to connect the button and LED properly to your board, for this to work properly. The LEDs and buttons will be connected to GPIO pins (General Purpose Input/Output pins) on your Beagle Board. You must have your LED connected to GPIO1\_17 and the button connected to GPIO3\_19. Instructions and significant help will also be provided in class and during lab sessions to assist you in learning how to connect the circuit properly. **Please make sure you take proper notes during these class sessions. Class instruction will likely be critical in assisting you in completing this assignment.** You will need to use the Pinout diagram below to determine how to connect your circuit to the appropriate pins.


**Step #1:** Connect a LED to GPIO49 and ground. Connect the Button to +3.3V and GPIO115.



**Schematic for Sample Program**

Notice that GPIO49 connects to pin 23 and GPIO115 connects to pin 27, as indicated in the pinout diagram below.

Beaglebone Black Pinout Diagram									
P9					P8				
Function	Physical Pins		Function		Function	Physical Pins		Function	
DGND	1	2	DGND		DGND	1	2	DGND	
VDD 3.3 V	3	4	VDD 3.3 V		MMC1_DAT6	3	4	MMC1_DAT7	
VDD 5V	5	6	VDD 5V		MMC1_DAT2	5	6	MMC1_DAT3	
SYS 5V	7	8	SYS 5V		GPIO 66	7	8	GPIO 67	
PWR_BTN	9	10	SYS_RESET		GPIO 69	9	10	GPIO 68	
UART4_RXD	11	12	GPIO_60		GPIO 45	11	12	GPIO 44	
UART4_TXD	13	14	EHRPWM1A		EHRPWM2B	13	14	GPIO 26	
GPIO_48	15	16	EHRPWM1B		GPIO 47	15	16	GPIO 46	
SPI0_CS0	17	18	SPI0_D1		GPIO 27	17	18	GPIO 65	
I2C2_SCL	19	20	I2C_SDA		EHRPWM2A	19	20	MMC1_CMD	
SPI0_DO	21	22	SPI0_SCLK		MMC1_CLK	21	22	MMC1_DAT5	
GPIO_49	23	24	UART1_TXD		MMC1_DAT4	23	24	MMC1_DAT1	
GPIO_117	25	26	UART1_RXD		MMC1_DATA0	25	26	GPIO_61	
GPIO_115	27	28	SP11_CS0		LCD_VSYNC	27	28	LCD_PCLK	
SP11_DO	29	30	GPIO_112		LCD_HSYNC	29	30	LCD_AC_BIAS	
SP11_SCLK	31	32	VDD_ADC		LCD_DATA14	31	32	LCD_DATA15	
AIN4	33	34	GND_ADC		LCD_DATA13	33	34	LCD_DATA11	
AIN6	35	36	AIN5		LCD_DATA12	35	36	LCD_DATA10	
AIN2	37	38	AIN3		LCD_DATA8	37	38	LCD_DATA9	
AIN0	39	40	AIN1		LCD_DATA6	39	40	LCD_DATA7	
GPIO_20	41	42	ECAPWMO		LCD_DATA4	41	42	LCD_DATA5	
DGND	43	44	DGND		LCD_DATA2	43	44	LCD_DATA3	
DGND	45	46	DGND		LCD_DATA0	45	46	LCD_DATA1	



LEGEND	
Power, Ground, Reset	
Digital Pins	
PWM Output	
1.8 Volt Analog Inputs	
Shared I2C Bus	
Reconfigurable Digital	

**Your Tasks:** Extending the sample code above, complete the following:

1. Connect an additional three LEDs to your BBB (for a total of four LEDs and one button). Help will be provided during class and lab session to assist you in connecting your circuit properly. You will need to acquire the LEDs, button and wires from the instructor or course TA.
2. Create a user-level program that passes a string containing 4 bits of information (for example "1001" or "0111"). The user-level program should continue to prompt the user until -1 is entered.
3. The driver should display the binary number on the four LEDs. For example, if the number "1111" is sent, all four LEDs should be turned ON. If the number "0111" is transmitted, only three LEDs should be turned ON.
4. When the button is pressed, it should act as a reset. In other words, when the button is pressed, all the LEDs should be turned OFF until another number is transmitted from the user space program.
5. Avoid race conditions. Two user space programs should never be able to have the driver open at the same time.
6. Make sure to edit the Makefile to build your userspace program.

**HINTS:**

- Look at the [user-level code from assignment #2](#). This should help you in constructing the user space program that sends data to the driver.
- Make sure to control the LEDs and button to a GPIO (General Purpose Input/Output) pin based on the pinout diagram.

**Submission Instructions:**

- Demonstrate that your code and circuit work to the TA or instructor during a lab session BEFORE the due date.
- Submit your code on Blackboard as a single tar file that contains (1) your driver code, (2) your user space code, and (3) a Makefile that compiles both programs.
- **Both the assignment demonstration and Blackboard code submission are required to receive credit for this assignment.**

**Grading Rubric**

Code submitted correctly. Contains Makefile, driver and user space program contained in a single .tar file on Blackboard.	10 points
Code is well organized, contains reasonable error checking and contains useful comments	10 points
User space code sends four bits of information to the driver and continues to prompt user for additional information until -1 is entered. Displays helpful and meaningful messages to the user.	10 points
When button is pressed, all LEDs are turned off by the driver	20 points
Driver turns LEDs on/off correctly as indicated by the user / user space program	30 points
Race conditions avoided. Driver does not allow two or more user space processes to access the driver at the same time.	20 points
<b>TOTAL</b>	<b>100 Points</b>

*\*Code that does not compile will not receive any credit*

*\*\*Your assignment must be demonstrated to a TA and submitted on Blackboard to receive credit*

**WARNING!** You may help each other, but your code submitted should not be copying significant portions of code. Submitted work needs to be your own. All programs will be screened for similarities using [MOSS software](#).