Project #4 – Missile Launcher Linux kernel driver

This project involves developing simple Linux device driver for the miniature missile launcher device designed for UAVs. The requirements for this project are simple.

First, you must wire up the missile launcher device to your Raspberry Pi. Please see Bob if you need assistance, although it should be straightforward via GPIOs as there are only four wires.

Second, you must construct a simple, character-based device driver to manage and control the missile launching device. While you might typically just employ user-space software for using GPIOs via sysfs when using Linux, you can also request and use the GPIOs within kernel modules.

Third, you must also provide an interface to read the status of and control your device that can be called from a user space application. For example, you can choose to leverage the syscall you did for the first project. You could also use the sysfs interface or the proc interface. It's your choice. However, your driver must minimally support four specific interfaces that can be called or used by a user space application:

- Your driver must provide an interface capable of firing a single missile. For example, call this interface, FIRE_ONE
- Your driver must provide an interface capable of rapidly firing all missiles.   For example, call this interface FIRE_ALL
- Your driver must support a read-only interface that tells you how many missiles remain to be fired. For example, call this interface NR_MISSILES_REMAINING
- Your driver must support a write-only interface that enables you to update the driver with the number of missiles currently loaded. For example, call this interface, SET_NR_MISSILES

Your driver must keep track of the number of missiles that are available to fire. You can do this simply be maintaining a counter that can be reset to the proper value after you've reloaded. You should be able to query your driver to determine how many missiles remain.

Finally, your driver must be self-contained and buildable outside the kernel tree. You must create a simple Makefile that references (and links to) the kernel you are running so you can test your driver. This will enable you to work "out of the kernel tree" and in your own directory. This makes rebuilding simple as you will not have to rebuild the kernel every time, only your driver.   You may use the default-installed kernel or you may use the 3.x linux kernel you downloaded and added a system call to during the first project. You driver must be able to be built using a Makefile! This is how most device drivers are bundled and distributed. Your Makefile need not be overly complex as many examples you'll find on the web. Simple is good.   Your driver must be able to be loaded into the kernel and removed from the kernel via the insmod/modprobe and rmmod commands respectively from the command line. Your driver should also log all operations performed into the system log (via printk) that is viewable using the dmesg command.

Please submit your final, functional driver code, Makefile, and a simple executive summary of the project, as well as how your configured your missile launcher to work on the Pi via Canvas.

Final note: You might find this tutorial useful as a reference: http://derekmolloy.ie/category/general/linux/ (Links to an external site.)Links to an external site.