Developing a low-level app on RaspberryPi with a SECURE kernel: writing an FM transmitter

In Linux, it is allowed to use the mmap() utility to map files or devices into the userspace (i.e. virtual address space of the calling process), potentially accessing them with r/w rights. On UNIX systems, /dev/mem is a character device file ((major, minor) = (1,1)) that is an image of the main memory of the computer. By creating by means of mmap() a new file descriptor with r/w mode pointing to the area of /dev/mem corresponding to the Peripheral Bus in the memory of a Raspberry Pi, one can control any of the low-level devices, such as the DMA controller, the GPIO clock controller or the Pulse Width Modulator, directly with reading and writing data onto the map.

Even though /dev/mem belongs to the superuser and has rights rw-------, it is potentially very dangerous to allow ANY user to read and write directly into physical memory with no constraint whatsoever. Modern BSD operating systems have a security policy that is based on the traditional 4.4BSD security model (see http://netbsd.gw.com/cgi-bin/man-cgi?secmodel_securelevel+9+NetBSD-5.0.1) that prevents _any_ user from writing to /dev/mem and /dev/kmem as soon as securelevel 1 ("secure mode"), which is the default securelevel in normal multi-user operation.

The purpose of this project is to write an FM transmitter using the onboard capacities of the Raspberry Pi (notably the PWM controller and the DMA), similarly to what has been done in unsecure mode by Oliver Mattos and Oskar Weigl (http://www.icrobotics.co.uk/wiki/index.php/Turning_the_Raspberry_Pi_Into_an_FM_Transmitter). Initial development will be for NetBSD but portability to other OSes running on the Raspberry Pi, including a GNU/Linux distribution (Raspbian, RISC OS, FreeBSD). Additionally to developing a functional, secure alternative to the code my Mattos&Weigl, there will be some investigation concerning the betterment of the audio quality.

Project highlights: operating systems, embedded programming, low-level programming, System-on-chip (SoC) programming, electronics, radiocommunications, signal processing. Student willing to take on this must have a good understanding of the different aspects of operating systems, must be able to read and use the documentation of an SoC and have some kind of interest in electronics and/or radiocommunications.

Programming language used: C.

For any queries or to apply, please contact me: Dr Jean-Baka Domelevo Entfellner (domelevo@gmail.com), CS dpt, room 1.24.