Emanuel Esquivel Lopez - 2016133597

Taller 8

Sistemas Embebidos

---

# Preguntas

## Drivers espacio usuario:

Estos son los drivers que en si no son los que interactúan directamente con el hardware, esto normalmente se realiza llamadas al sistema operativo, este ultimo es el encargado de realizar la interacción con el dispositivo.

Estas llamadas pueden ser para lectura o escritura, además funcionan como interfaz entre el espacio user – y espacio kernel.

El kernel es el encargado de llevar las peticiones del usuario al hardware.

```c
int exPin(int pin){
    const int BUFFER_MAX = 3;
    char buffer[BUFFER_MAX];

    ssize_t bytesSize;
    int file;

    file = open("/sys/class/gpio/export", O_WRONLY);
    if (file == -1){
        fprintf(stderr, "Filed to open");
        return -1;
    }

    bytesSize = snprintf(buffer, BUFFER_MAX, "%d", pin);
    write(file, buffer, bytesSize);
    close(file);
}
```

Figura 1: Ejemplo driver usuario.

En este caso podemos ver que debemos llamar al sistema para escribir en la GPIO, llamando la función call, la cual debe escribir el archivo de la ruta especificada de manera que no interactúan directamente con el hardware.

## Drives espacio kernel

Estos son los encargados directamente con la comunicación principal del hardware, además de que con esto nos ahorramos las llamadas al sistema anteriormente utilizadas, si el usuario desea utilizar estos debe usarse mediante polling, para poder proveer una interfaz entre el kernel y el user.

```
/* Declaration of mem.c functions */
int memory_open(struct inode *inode, struct file *filp);
int memory_release(struct inode *inode, struct file *filp);
ssize_t memory_read(struct file *flip, char *buf, size_t count, loff_t *f_pos);
ssize_t memory_write(struct file *flip, const char *buf, size_t count, loff_t *f_pos
);
int memory_init(void);
void memory_exit(void);

struct file_operations memory_fops = {
  read : memory_read,
  write : memory_write,
  open : memory_open,
  release : memory_release
};

/* Declaration of the init and exit functions */
module_init(memory_init);
module_exit(memory_exit);
```

Figura 2: Ejemplo driver kernel.

Con esto definimos las funciones como tal para que se interactúa con el hardware quitando así las llamadas al sistema.