|  |  |  |
| --- | --- | --- |
| **Archivo** | **Subrutinas** | **Porcentaje de Código** |
| Gpio0\_2.s | **GetGpioAddress**: obtiene la dirección virtual de memoria de los puertos GPIO de la Raspberry.  Entradas: -  Salidas:   * R0: retorna la dirección de los GPIO   **SetGpioFunction**: establece si un puerto GPIO va a ser usado como lectura o como escritura.  Entradas:   * R0: número de pin * R1: la función a realizar, 0 lectura, 1 para escritura.   Salidas: -  **SetGpio:** estableces como encendido o apagado un GPIO específico  **Entradas:**   * **R0:** número de pin * **R1:** 0 para pagar el pin, 1 para encenderlo   **Salidas: -**  **GetGpio:**  **Entradas:**   * **r0:** número de puerto   **Salidas:**   * **r0:** 1 si el puerto está en high, 0 de lo contrario | 171 líneas = 12.28% |
| phys\_to\_virt.c | **phys\_to\_virt:** permite modificar la memoria protegida de la raspberry.  **Entradas:** -  **Salidas:**  R0: dirección virtual de la memoria de la raspberry | 30 líneas = 2.15% |
| pixelV2.c | **getScreenAddr**: devuelve la dirección virtual de memoria de video para trabajar con ella en programas ARM.  **Entradas:** -  **Salidas:**   * **r0:** puntero a la dirección de memoria.   **getScreenXSize:** devuelve en r0 el tamaño de panta en X.  **getScreenYSize:** devuelve en r0 el tamaño de pantalla en Y. | 67 líneas = 4.81% |
| timeLibV2.c | **better\_sleep:** método que realiza una pausa exacta en cantidad de segundos o milisegundos.  **Entrada:**   * r0: cantidad de segundos para la pausa   **Salida: -** | 51 líneas = 3.66% |
| Convert.py | Archivo utilizado para generar las matrices de pixeles que utiliza ARM para pintar en la pantalla. | 41 líneas = 2.94% |
| libreriasHtml.c | **htmlUpdaterA210, htmlUpdaterA211, htmlUpdaterA212:** corren un comando del bash de LINUX, que permite ejecutar un script de python con argumentos predefinidos.  Entrada:   * r0: 1 si el salón está disponible 0 s si no lo está.   **Salida: -** | 57 líneas = 4.09% |
| status.py | **Actualización** método principal que es llamado desde C.  Entrada   * **aulaPram:** recibe el numero del salón que se va a editar * **condicionParam:** recibe si el salón está disponible o n**o**   **Salida: -**  **actualizarEstadoAula1, actualizarEstadoAula2, actualizarEstadoAula3:** llama a updateHTML  **Entrada**   * **entrada** recibe el esto del salón.   **Salida: -**  **UpdateHTML:** modifica el archivo html que se despliega como pagina web  **Entrada**   * **Salon** * **Estodo: ocupado o disponible**   **Salida: -** | 129 líneas = 9.26% |
| estado.html | **Código html que se muestra como página web. Este es modificado por medio del script de python.** | 22 líneas = 1.58% |
| Métodos.s | **welcomeImg:** imagen de presentación  **welcomeImg2:** imagen con código QR  **blackScreenImg:** imagen que limpia la pantalla pintando todo de negro  **a211LibreImg, a211OcupadoImg, a210LibreImg, a210OcupadoImg, a212LibreImg, a212OcupadoImg:** cada una se utiliza para pintar el estado del salón según sea indicado en el min.  **Entradas: - ,** existe una variable global en memoria que indica el origen en x,y de cada imagen.  **Salidas: -** | 456 líneas = 32.74% |
| Main.s | **welcomeLoop:** pinta imagen de inicio, revisa el botón start.  **InfoPage:** muestra la imagen con el codigo QR  **checkButtons:** revisa si los swiches han sido cambiados. llama a la función de update respectiva de cada salón.  **a211L, a211O, a210L, a210O, a212L, a212O:** funciones de update que actualizan los LEDs, las variables que se leen para pintar las imágenes referentes al estado de los salones y el archivo html.  **printResults:** limpia la pantalla. Luego despliega la imagen con el número de salón y estado. Rojo si está ocupado, verde si está libre.  **Entradas: -**  **Salidas: -** | 369 líneas = 26.49% |

todas las líneas = **996 (ARM) + 397 (Alto nivel) = 1393**

Líneas de Matrices = **2950**