| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| / | | | | |
| | configuracion | data : binario | [No se puede abrir] | |
| | connectNetwork | script bash | Inicializa wvdial | wvdial<br>/var/slr/3g.wvdial.conf<br>/var/slr/network.script<br><br>Los archivos de /var/slr están en Vol50MB |
| | construye | script bash | Ejecuta los make de cada directorio (módulos) en cascada | |
| | Makefile | Makefile | Compila e instala el sistema.<br><br>Para la compilación se llama a ./construye<br>Para la instalación, se copian los ejecutables recién compilados a /bin | |
| | preparatoria | script sh | - Remonta el sistema de archivos en modo lectura-escritura.<br>- Ejecuta el Makefile<br>- Sincroniza el sistema de archivos con sync<br>- Pone el sistema en hora con ntpdate<br>- Ejecuta utiles/inicializador<br>- Comprueba si /lms se ha marcado para borrar, en tal caso, elimina la carpeta y todos sus archivos.<br>- Sicnroniza el sistema de archivos con sync<br>- Reinicia el equipo con init 6 | mount -n -o remount, rw /<br>init 6<br>sync |
| | setAsNew | script bash | Resetea el sistema al estado de fábrica.<br><br>- Ejecuta utiles/inicializador<br>- Limpiar los logs en (/var/slr/logs.serial)<br>- Resetea el fichero de datos `configuracion` y `configuracion.default`<br>- Limpia el registro de reinicios del sistema (./restarts)<br>- Elimina calibracionReference<br>- Elimina firstLogin | |
| **comun/** | | | | |
| | aes.cpp | cpp | Anesma, an AES encryption library for C++ | https://es.wikipedia.org/wiki/Advanced_Encryption_Standard |
| | aes.h | h | Cabeceras del algoritmo de encriptado AES | Tanto el .h como el .cpp son código externo (open soruce) por lo que está debidamente documentado. |
| | Calibracion.cpp | cpp | Controla la calibración del limitador. Controla los 3 propiedades:<br>- referencia: por defecto 0. Se inicializa como ref², siendo ref el valor pasado en el constructor. Se usa para calcular los dB.<br>- ganancia: no se actúa sobre el, solo se inicializa en el construtory se devuelve en el getter<br>- db_en_referencia: por defecto 0. Se usa para calcular los dB. | https://es.wikipedia.org/wiki/Decibelio#Aplicaciones_en_telecomunicaci%C3%B3n |
| | Calibracion.h | h | Definición de la clase Calibración. Define los métodos y atributos de clase. | Incluye:<br>- Config.h |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | compact.cc | cc | Contiene una serie de funciones las cuales manejan cadenas de texto. La finalidad de las funciones es encriptar cadenas de texto. La cadena encriptada es el doble de grande que la cadena actual. Para cada caracter de la cadena original, se generan dos nuevos caracteres correspondientes a los bits más y menos significativos de la representación en código ASCII del caracter.<br><br>Ver código adjunto | Incluye:<br>- Configuracion.h<br>- EstadoDelLimitador.h |
| **Importante** | Config.h | h | Contiene exclusivamente definiciones (defines). Es por tanto un almacén de constantes globales y valores por defecto que se usan a lo largo del código del limitador. | Incluye: sys/soundcard.h<br>PuertoDeEstado: 9099 **¿?**<br>GananciaPorDefecto: 50 |
| | Configuracion.cpp | cpp | Implementación de algunos de los métodos de las clases definidas en el .h<br><br>Unos cuantos métodos tienen el comentario: "Cambios por la ampliación de la normativa semanal y festivos"<br>Todos estos métodos tienen la finalidad de gestionar la configuración por normativa extendida, es decir, la configuración según día de la semana, festivo u hora del día.<br><br>Hay un par de métodos para encriptar/desencriptar, y que se usan al guardar/cargar la configuración hacia/desde archivo -> "var/srl/configuracion"<br><br>- Método crc() -> Devuelve un entero igual a la suma de todos los atributos de configuración (algo así como un checksum)<br>- Método sonometro() -> lee 80 caracteres de /dev/dsp y los devuelve. | dsp, crc<br><br>PuertoDeConexion: 9182 **¿?** |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| ABOMINACIÓN! | Configuracion.h | h | Controla todo lo que es la configuración del limitador, como los parámetros de configuración, datos del local, del distribuidos, del servidor, incluyendo la configuración específica por franja horaria, día de la semana, festivos y demás (la normativa).<br><br>Contiene varias clases e incluso la implementacion de algunos de sus métodos en el mismo archivo de cabecera (de ahí lo de abominación). En concreto, este archivo define las siguientes clases:<br>- Intervalo<br>- Normativa<br>- Festivo<br>- Configuración<br><br>E incluye la implementación de los métodos de las 3 primeras.<br>- Intervalo:<br>- Atributos: horaInicio y horaFin<br>- Métodos: un constructor y un método dentro() que comprueba si la hora dada está dentro de ese intervalo (hora_inicio < hora < hora_fin)<br>- Normativa:<br>- Atributos: un Intervalo y 4 floats: maximoNocturno, maximoDiurno, máximoRecepcionNocturno, máximoRecepcionDiurno<br>- Métodos: dos constructores, uno por defecto y otro con argumentos. Por defecto el intervalo es de 8 a 22h, con un maximo diurno y nocturno de 95dB. 2 funciones que devuelven el maximo y maximoRecepcion dependiendo del horario (intervalo.dentro(h))<br>- Festivo: es la clase más extensa<br>- En general, contiene los atributos propios de un día (todos int o float), hora de inciio, horas de duración y el maximo de emision y recepción para ese día festivo. Los métodos manejan fechas y horas del tipo struct tm y time_t | Incluye:<br>- Config.h |
| | Configurador.cpp | cpp | Implementación de la clase.<br><br>- limpiaCadena: reemplaza todos los '^' por espacios.<br>- escribeConfiguración: escribe la Configuración en el archivo especificado como argumento.<br>- configuraDeFichero: abre el fichero dado en modo lectura.<br>- configuraDeFlujo: invocado desde configuraDeFichero, lee el fichero en trozos de 200 caracteres, y va actualizando los atribitos del objecto Configuración según los va encontrando en el fichero. Finalmente, actualiza la Configuración con el método graba() de su clase. | Para el modo comprimido, usa:<br><br>gzip -dc <fichero><br><br>Archivo de Configuración -><br>/var/slr/configuracion |
| | Configurador.h | h | Definición de la clase con mismo nombre. Solo tiene métodos. | Incluye:<br>- Configurador.h<br>- Configuracion.h |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | EstadoDelLimitador.cpp | cpp | Implementación de la clase.<br><br>Solo existe un método: actualiza(), el cual abre el archivo de estado en modo lectura y comprueba si el estado ha cambiado o no. No modifica ningún atributo de la clase, por lo que no queda claro porque se ha nombrado como "actualiza".<br><br>El método devuelve bool, por lo que se intuye que el método devuelve si el estado debe ser actualizado o no, pero no se sabe quién se encarga de dicha actualización (debería ser la propia clase la que cambie su estado...) | Incluye:<br>- Config.h<br>- EstadoDelLimitador.h |
| | EstadoDelLimitador.h | h | Definición de la clase con el mismo nombre. Define las propiedades y el estado del limitador como atributos:<br>- Número de serie<br>- Versión<br>- Penalización, atenuanción y atenuación global<br>- Presión, presión media, presión izquierda y presión derecha<br>- Hora del día.<br>- Nivel de recepción y nivel máximo.<br>- Micrófono conectado | Se almacena en el fichero -> /tmp/estado.slr |
| | EstadoDeModificadores.cpp | cpp | Implementación de la clase.<br><br>5 métodos: constructor (ambos atributos a false), getters, guarda y lee. | Incluye:<br>- Config.h<br>- EstadoDeModificadores.h<br><br>F_Llave -> /tmp/slr.k |
| | EstadoDeModificadores.h | h | Definición de la clase con mismo nombre.<br><br>Dos atributos de tipo bool: inhibidor y microfono | |
| | Funciones.cpp | cpp | Funciones varias, algunas bastante interesantes para exportar el estado completo del limitador a distintos formatos.<br><br>- volcado<br>- volcadoJSON<br>- volcadoXML<br>- volcadoYML<br>- volcadoSQL (2)<br>- volcadoSQLite<br>- imprime<br>- generaGrafico<br>- generaInforme<br>- monta<br>- desmonta<br>- configura<br>- configuraDeFichero<br>- limpia<br>- guarda<br>- informeEncendido<br>- informeDeConfiguracion<br>- generador | Incluye:<br>- Funciones.h<br>- Configurador.h<br>- limitador/Registrador.cc<br>- limitador/Calibración.cpp<br>- Serial.cpp<br>- Serial.h<br><br>Ruta de archivos: /tmp |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | Funciones.h | h | Defincición de la clase con mismo nombre. Encapsula una serie de funciones de uso general, ya que manipula o usa objetos del tipo Configuracion EstadoDelLimitador y Registro | Incluye:<br>- Configuración.h<br>- limitador/Registro.h<br>- EstadoDelLimitador.h<br>- Config.h |
| | getSerial.cpp | cpp | Parece que solo es un fichero de prueba para testear Serial.cpp<br><br>Usa las funciones getID, testID y printDefineSerial de Serial.cpp | Incluye:<br>- Serial.cpp |
| | gSerial | ejecutable | Obtiene el ID (HWaddr de eth) y lo escribe como un define en Serial.h | |
| | Makefile | Makefile | Contruye el ejecutable gSerial a partir de getSerial.cpp, luego lo ejecuta y redirige su salida a Serial.h | |
| | Serial.cpp | cpp | Varias funciones realizacionadas con el serial (ID) del equipo. El ID se presupone único para cada equipo.<br><br>- getID: genera el ID a partir de los dos fd (ver siguiente celda), los concatena, ( fd2 + fd1) y da la vuelta al string resultante, para finalmente devolverlo en id (puntero a char dado como parámetro)<br>- testID: comprueba si el ID es igual al ID dado como parámetro<br>- strip: *creo* que elimina los espacios en blanco de la cadena de texto dada como parametro<br>- getHSerial: consulta y devuelve la identidad del controlador IDE maestro del sistema y devuelve los primeros 20 caracteres en out (puntero a char pasado como parámetro en la llamada a la función.<br>- printDefineSerial: imprime un define para el ID (#define id <id>)<br>- maskString: recibe in y out, realiza una encriptación sobre cada caracter de in y lo almacena en out | El ID es generado a partir de los siguientes file-descriptors<br><br>fd1 -> ifconfig \| grep eth \| grep HWaddr<br>fd2 -> /dev/hda<br><br>ioctl, strndup |
| | serial.h | h | Vacío. gSerial redirije su salida a este archivo. Esto ocurre al ejecutarse el Makefile de este directorio. | |
| | TestConfig.cpp | cpp | Fichero de prueba de Configuración.cpp | |
| **comunicacion.old/** | | | | |
| | cpp | | | |
| | old | | | |
| | a.out | | | |
| | cliente.c | | | |
| | funciones_Conexion.cpp | | | |
| | funciones_envioDeDatos.cpp | | | |
| | funciones_formato.cpp | | | |
| | Funciones.h | | Reemplazado por comv2 | |
| | funciones_hora.cpp | | | |
| | globales.h | | | |
| | Main.cpp | | | |
| | Makefile | | | |
| | slrComm | | | |
| | Test.cpp | | | |
| | TestRegistrador.cpp | | | |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| **comv2/** | | | | |
| | cpp | directorio | Contiene los ficheros de un proyecto que implementa un Socket en C++. Contiene el fichero de cabecera para el socket y otra para sus excepciones, la implementación del servidor, la implementación del cliente, una demo, y un Makefile para compilar el proyecto. | |
| | old | directorio | | |
| | base64.c | c | Funciones para codificar y decodificar char a base64 | |
| | cliente.c | c | Se conecta a un servidor remoto en un puerto determinado . Si no puede devuelve -1 | |
| | encrypter.cpp | cpp | Contiene varias funciones para encriptar y desencriptar strings | |
| | funciones_Conexion.cpp | cpp | Implementación de la mayoría de las funciones definidas en funciones.h, salvo las de envio de datos que se implementan en funciones_envioDeDatos.cpp. | Incluye:<br>- funciones.h<br>- cliente.c<br>- encrypter.cpp<br>- WhiteRabbit.h<br>- comv2/cpp/Socket.{h, .cpp}<br>- comun/serial.h |
| | funciones_envioDeDatos.cpp | cpp | contiene 2 funciones para enviar información al servidor (data.boanergesnetwork.com). Una se encarga de enviar la configuración del sistema, la otra se encarga de enviar registros (logs) del limitador | incluye funciones.h globales.h |
| | funciones_formato.cpp | cpp | Funciones para generar cadenas de texto en un formato predefinido. Las cadenas de texto van a ser posteriormente enviadas a un servidor, el cual espera estos formatos. | sprintf(char *str, char *format, *opts)<br>Fomatea el string str siguiendo el formato dado en fomat, como se haría con printf, pero en lugar de imprimir el resultado, lo almacena (machaca) en str.<br><br>incluye funciones.h |
| | Funciones.h | h | Definición de funciones relacionadas con el envío / recepción de mensajes al servidor configurado (imagino que es el servidor del Ayuntamiento). | Destacable el hecho de que las funciones están bastante bien comentadas. Se indica la descripción de la función, parámetros que recibe, precondiciones y valor revuelto. Además, en la mayoría de los casos se indica también el fichero en el que se encuentra la implementación. |
| | funciones_hora.cpp | cpp | Contiene funciones para manejar la hora y fecha en varios tipos de estructuras de datos (tm, time_t, char). Las funciones reciben la hora en formato "yy/mm/dd hh:mm:ss".<br><br>Contiene una función 'actualizarHora(char *hora)', la cual intenta actualizar la hora del sistema a la hora recibida como parámetro, sin embargo, el código para actualizar la hora del sistema está comentado y la función solo imprime por pantalla que no se cambiará la hora del sistema por ser peligroso. | mktime, sscanf, strftime<br><br>incluye funciones.h |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | globales.h | h | Definición de funciones y macros.<br><br>Hay definidas 2 macros de configuración del sistema que indican la frecuencia con la que el sistema actualiza la hora (cada 5 días, si no se reinicia antes) y la configuración (re-carga de archivos de configuración imagino; cada hora) | #define FicheroDeMarcaSinConexion /tmp/nonAvailableConnection<br><br>incluye otras cabeceras:<br>- limitador/registro.h<br>- comun/configuracion.h<br>- limitador/Calbiracion.h |
| | Main.cpp | cpp | Código principal (main) del programa ejecutable srlComm. Se comunica con un servidor en la dirección data.boanergesnetwork.com:3002 , al cual envia información. Este porgrama realiza varias funciones, pero principalmente intenta pasar por un shell remoto. Puede recibir acciones como getConfig, getEvents, getRegistry, etc. Ejecuta la ción requerida y envía el resultado de la operación. | |
| | Makefile | Makefile | Genera el ejecutable slrComm<br><br>Dependencias:<br>- Internas:<br>- Main.cpp<br>- funciones_hora.cpp<br>- funciones_formato.cpp<br>- cliente.c<br>- Externas:<br>- comun/configuracion.cpp<br>- limitador/Registro.cpp<br>- limitador/Estado | Opciones gcc:<br>* -lm: link math lib<br>* -lpthread: link thread.h<br>* -Wno-write-strings: suppress warnings |
| | slrComm | ejecutable | | |
| | Test.cpp | cpp | Prueba el módulo conectándose a localhost:8081 | Incluye cliente.c |
| | TestRegistrador.cpp | cpp | Realiza un test sobre cada uno de los registros alamcenados en el sistema. Al parecer existe un registro para que almacena el estado del sistema en cada instante de tiempo (todo el texto que inunda la pantalla son estos registros) | ctime: convert time_t value to string<br><br>Incluye:<br>- limitador/Registrador.cc<br>- limitador/Registro.cpp<br>- limitador/Estado.cpp<br>- comun/Configuracion.cpp |
| | WhiteRabbit.cpp | cpp | Implementación del algoritmo de encriptado Rabbit | |
| | WhiteRabbit.h | h | Definición de la clase WhiteRabbit. Rabbit es un algoritmo de encriptado | https://en.wikipedia.org/wiki/Rabbit_(cipher) |
| | WhiteRabbit.o | o | - | |
| | WhiteRabbit_Original_AllInOneFIle.cpp | cpp | Implementación del algoritmo de encriptado Rabbit. Definición e implementación. | |
| **instalador/** | | | | |
| | fdiskCommands | ??? | ??? | |
| | lm701f.localCommands | ??? | ??? | |
| | MAKEDEV | script sh | "A very simple script that can be used to create some of the more common device nodes found in /dev." | https://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/dev.html |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | sdbInstall | script bash | Inicializa la estructura de directorios principal del sistema de archivos:<br><br>- dev<br>- proc<br><br>TODO | mkfs.ext3, fdisk, sed, mknod |
| | sdcInstall | script bash | Inicializa la estructura de directorios principal del sistema de archivos:<br><br>- dev<br>- proc<br><br>TODO | mkfs.ext3, fdisk, sed, mknod |
| **limitador/** | | | | |
| | 1 | config file | Archivo de configuración de alsaplayer | |
| | a.out | ejecutable | | |
| | Atenuador.cpp | cpp | Clase Atenuador.<br><br>Contiene un file descriptor a cada mixer, los cuales vienen definidos en el archivo "mixers", y los abre en modo lectura-escritura.<br>El método atenuación se encarga de calcular los valores necesarios y escribir (actualizar) los mixers, usando MIXER_WRITE | Incluye:<br>- sys/soundcard.h<br>- mixers<br><br>Comentario:<br>Para el control de varios ambientes esto debe cambiar por el dispositivo que haya en la configuracion y deben existir tantos atenuadores como mezcladores<br>Otra opción es crear tantos atenuadores como mezcladores y asignar a cada canal su mezclador.<br>Ej:<br>/dev/dsp -> /dev/mixer<br>/dev/dsp1 -> /dev/mixer1 |
| | AtenuadorMk163.cpp | cpp | La clase no se usa en ninguna parte. | Un comentario en el código dice lo siguiente: "Ya posee los cambios necesarios para trabajar con el atenuador DS"<br><br>Incluye:<br>- PuertoParalelo.cpp |
| Este es el atenuador que se usa en el limitador | AtenuadorPga.cpp | cpp | | |
| | AtenuadorPga.h | h | Declaración de la clase AtenuadorPga.<br><br>Contiene varias constantes (defines) para identificar los pines del puerto paralelo sobre los que comunicarse, varios float para controlar el nivel de atenuación y una instancia de PuertoParalelo, para poder leer/escribir sobre los pines del puerto paralelo. | Incluye:<br>- comun/config.h<br>- PuertoParalelo.cpp |
| | audioOrder | script sh | Re-monta los módulos kernel de sonido:<br>- snd_usb_audio<br>- snd_cs5535audio | rmmod, modprobe |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | Calibracion.cpp | cpp | Implementación de la clase Calibración.<br><br>El calculo de los decibelios se realiza comparando la potencia leída con la potencia de referencia, según la formula que puede verse aquí. | Datos de calibración a fichero F_KAL<br><br>Incluye:<br>- Calibracion.h |
| | Calibracion.h | h | Definición de la clase Calibracion.<br>Atributos:<br>- dbRef, ref, ruido, ganancia, equilizacion<br><br>Métodos:<br>- leerCalibracion<br>- guardarCalibracion<br>- calculaBs<br>- calculaEnergia<br>- print | Incluye:<br>- ../comun/Config.h |
| | CalibrationTest.cpp | cpp | Incompleto. No se usa. | |
| | Estado.cpp | cpp | Implementación de los métodos de la clase Estado. Son todos setter y getters.<br><br>Para la gestión de los informes se utiliza una máscara de bits, por lo que las operaciones sobre el atributo 'informe' se realizan utilizando operadores a nivel de bit. | Incluye:<br>- Estado.h<br>- ../Config.h |
| | Estado.h | h | Define los atributos para representar el estado del limitador, así como métodos para manejarlos.<br><br>Lo más destacables son:<br>- atenuacion, penalización y máximo<br>- presion, presionIzquierda, presionDerecha y recepcion | |
| | filterTest.cpp | cpp | Despreciable | |
| | filtroA | directorio | Contiene filtroA.cpp, una implemetación descargada de internet para el filtro IIR | http://www.winfilter.20m.com |
| | iir.cpp | cpp | Clase que implementa el filtro IIR.<br><br>Atributos:<br>- orden : int<br>- estado : double<br>- numerador : doube<br>- denominador : double<br><br>Métodos:<br>- constructor()<br>- setIIR(int orden) --> estado\|numerador\|denominador[orden+1] = 0.0<br>- loadIIR(stream archivo)<br>- showIIR(stream archivo)<br>- IIRfilter(short sample) : double | Fichero de datos: FilterBankFile |
| | LectorAlternativo.cpp | cpp | No se usa (solamente se usa en Sonometros.cpp pero está comentado) | Incluye:<br>- octaveBandFilter.c |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | Lector.cpp | cpp | Contiene un par de funciones al principio autogeneradas por el generador de código para filtros digitales **mkfilter**. La orden utilizada para la generación de dichas funciones están comentadas al comienzo del fichero.<br><br>Clase Lector. Utiliza tipos IIR, qiir y Lectura2C.<br><br>Atributos:<br>- sd<br>- buffer[muestrasALeer * canales]<br>- frec<br>- nSig<br>- canales<br>- applyAWeight<br>- IIR filtroIzquierda, filtroDerecha<br><br>Métodos:<br>- aplicaFiltrado: por cada filtro, recorre el buffer y obtiene su media positiva.<br>- inicializarFiltros: lee los filtros desde FilterBankFile<br>- analiza: inicializa y aplica los filtros mediante los métodos qiir_init y qiir_filter de la clase qiir<br>- fijaPartesDeLectura(int n): nSig = frec / n<br>- reOpen(fichero, nCanales): abre /dev/dsp (ioctl)<br>- aplicaCorreccion0Logico: obtiene la media de los valores de buffer[] y se la resta a cada uno de los valores de buffer[]<br>- lectura() : Lectura2C --> devuelve una salida de tipo Lectura2C, de la cual solo se inician los valores isMono y applyInternalCorrection (ambos de tipo lógico) dependiendo de si la señal es mono o no. Aplica los filtros a los canales usando aplicaFiltrado | https://github.com/sven337/mkfilter/blob/master/gencode.C<br><br>dsp --> /dev/dsp |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | Lectura2C.cpp | cpp | Clase Lectura2C: lectura de 2 canales (izquierda-derecha). Realiza la función de sensores para las salidas, de forma que es capaz de medir, procesar y almacenar las señales de salida en sus dos canales.<br><br>Define 3 vectores con parámetros para los filtros (FrecuenciaCentral, FiltroA, FiltroDeAjuste).<br>Para almacenar el estado de los canales, utiliza las siguientes propiedades:<br>- energial[8] : int<br>- energiaD[8] : int<br>- dBI[8] : float<br>- dBD[8] : float<br>- dBAI[8] : float<br>- dBAD[8] : float<br><br>En cuanto a métodos. contiene los siguientes:<br>- sum(Lectura2C l): suma de dos Lectura2C<br>- div(float x): divide cada propiedad mencionada anteriormente entre x<br>- zero(): pone todo a 0 \| constructor<br>- print(): imprime por pantalla<br>- valorGlobalEspectro(float dbs) : float --> calcula y devuelve el resultado de aplicar la formula que se ve a la derecha (**decibelios ponderados**)<br>- globalIzquierdoPonderado(): devuelve valorGlobalEspectro(dBAI)<br>- globaDerechoPonderado(): devuelve valorGlobalEspectro(dBAD)<br>- globalIzquierdoPonderadoDeRecepcion y globalDerechoPonderadoDeRecepcion(): recibe un vector de float (aislamiento), y los resta a dBAI, dBAD (según el caso) si y solo si aislamiento[i]<3<br>- save(int n): vuelca la memoria de la instancia actual en el fichero /tmp/.lx[n]<br>- read(int n): aplica el vuelco de memoria del acrhvio /tmp/.lx[n] a la instancia acctual | Incluye:<br>- Calibracion.h<br>- qiir/qirr.c<br>- qirr/datosSOS_fbank_scaled.h<br><br>Archivo de sensor -> F_Lectura<br><br>Niveles ponderados |
| | limitador | ejecutable | | |
| | Limitador.cpp | cpp | | Incluye:<br>- Atenuador.cpp<br>- SevidorDeEstado.cpp<br>- comun/Configuracion.h<br>- comun/EstadoDeModificadores.h<br>- Registro.h<br>- Registrador.cc<br>- sys/io.h<br>- AtenuadorPga.h<br>- Sonometros.cpp<br><br>En la salida estándar, aparece una línea que dice <!-- Reopenning --> cada 4 salidas normales (registros). ¿Qué significa? |
| | Makefile | makefile | Construye el ejecutable 'limitador' | |
| | mixers | C/C++ | Define un vector con el path a 3 mixers:<br>/dev/mixer1, /dev/mixer2, /dev/mixer3 | Se incluye en Atenuador.cpp |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | octaveBandFilter.c | c | Conjunto de filtros (funciones) autogenerados por **mkfilter**. Las órdenes que se utilizaron para generarlos se encuentran comentadas en las cabeceras de cada función. | |
| | PruebaAtenuador | ejecutable | | |
| | PuertoParalelo.cpp | cpp | Define e implementa la clase PuertoParalelo, la cual permite leer y escribir en los registros de datos del puerto paralelo de la máquina.<br><br>El registro de datos del puerto o bus paralelo se compone de 8 bits bidireccionales en la direccion 0x378. Pines de datos [2-9] | io.h<br>https://es.wikipedia.org/wiki/Puerto_paralelo<br>http://www6.uniovi.es/cscene/CS4/CS4-02.html |
| | qiir | directorio | Contiene la implementación de un filtro digital en casdada en Forma Directa II Traspuesta. Los archivos más importantes son qiir.h y qiir.c, ambos comentados y bastantes limpios en cuanto a código. Esta clase se incluye en la compilación de **Limitador** | Comentario sobre la cabecera de la función qiir_filter(quiir *f, int x):<br>"Calcula la salida para la entrada x utilizando una **Casdada en Forma Directa II Traspuesta Proakis, Manolakis. "Digital Signal Processing" 4th Ed, pg 589**" |
| | Registrador.cc | cc | Implementa los métodos de la clase Cabecera, y define e implementa la clase Registrador.<br><br>En la clase Cabecera, el contructor incializa sus atributos a valores estáticos, los cuales son:<br>- tipo=370<br>- horaBase=time(NULL)<br>- cadena=" rfs ( Acusticayaudio.com ) "<br><br>La clase Registrador se encarga de las operaciones de gestión de los Registros, mayormente lectura y escritura.<br><br>- inicializa(): inicializa el fichero F_Registro, con una cabecera y 1000 registros vacíos.<br>- horaBase(): lee la hora base desde la cabecera del archivo F_Registro<br>- obtenPosicion(hora): devuelve la posición del registro de la hora pasada como argumento<br>- guardarRegistro(registro r): escribe el registro r en la posición que le corresponde (según su hora)<br>- leeRegistro(): varias implementaciones, pero todas leen el registro dado desde F_Registro y lo almacena en un registro R.<br>- pasaRegistros(): lee una franja de registros, desde time_t inicio a time_t fin. Los escribe en F_SalidaRegistro | Incluye:<br>- Registro.h<br><br>Archivo de datos: F_Registro, F_SalidaRegistro |
| | Registro.cpp | cpp | Implementación de los métodos de la clase Registro.<br><br>La mayoría de los métodos son getters. Los métodos para devolver las presiones y la atenuacion_maxima devuelven los valores divididos por 2. | Incluye:<br>- comun/Config.h<br>- Registro.h<br><br>Archivo de datos: F_Registro |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | Registro.h | h | Contiene dos clases, Cabecera y Registro.<br><br>Registro consiste en una serie de atributos tomados de Estado y Configuración, y una lista de métodos para manejar dichos atributos y leer/escribir el registro a disco.<br><br>Atributos:<br>- hora<br>- informe<br>- atenuacion, atenuacion_max, db<br>- crc<br>- presion_Izquierda, presion_Derecha, presion_Rececpcion | Incluye:<br>- Estado.h<br>- comun/Configuracion.h" |
| | ServidorDeEstado.cpp | cpp | Implementación de la clase ServidorDeEstado. Sendos métodos para escribir y leer el estado del limitador hacia/desde el fichero F_Estado. | Incluye:<br>- ServidorDeEstado.h<br><br>Fichero de datos: F_Estado |
| | ServidorDeEstado.h | h | Definición de la clase. Contiene un atributo del tipo EstadoDelLimitador, y dos métodos para leer y actualizar el estado. | Incluye:<br>- Estado.h<br>- comun/EstadoDelLimitador.h<br>- comun/Configuracion.h |
| | sonometro | ejecutable | | |
| | Sonometro.cpp | cpp | Solo contiene un main vacío. | Incluye:<br>- Sonometros.cpp |
| | sonometros | ejecutable | | |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | Sonometros.cpp | cpp | Clase Sonometro<br><br>Atributos:<br>- int _lecturasPorSegundo<br>- Lector *microfono<br>- Lector *lineas<br>- Calibracion calibracionMic<br>- Calibracion calibracionLi<br>- Calibracion calibracionLd<br>- Lectura2C ultimaLecturaDeMicrofono<br>- Lectura2C ultimaLecturaDeLineas<br><br>Métodos:<br>- Sonometros(char *tarjetaMic, char *tarjetaLinea): incializa los lectores (sensores) para el microfono y las líneas (audio). El segundo puede ser NULL, de forma que el limitador pueda actuar en modo registrador (solo registra, no limita "CREO:<br>- reloadCalibracion: fuerza el refresco de calibracionMic, calibracionLi, calibracionLd<br>- numeroDeLecturasPorSegundo(int n): _lecturasPorSegundo = min(n, 10). _lecturasPorSegundo -> microfono y líneas<br>- startLines(): crea y lanza una hebra para el control de las lineas. Ejecuta la función Sonometros_auxFunction<br>- Sonometros_auxFunction: bucle infinito. Realiza lo mismo que read pero para la líneas (audio) en lugar de para el mcirofono<br>- read(): actualiza ultimaLecturaDeMicrofono, invocando a la funcion lectura de la clase Lectura2C. Renicia las lecturas cuando se alcanzan las lecturas por segundo. | Incluye:<br>- sys/soundcard.h<br>- sys/iotcl.h<br>- Lectura2c.cpp<br>- Calibracion.cpp<br>- iir.cpp<br>- Lector.cpp<br>- LectorAlternativo.cpp (comentado)<br><br>Fichero de datos: F_Kal<br><br>Micrófono --> /dev/dsp1<br>Audio --> /dev/dsp |
| | SwitchPink.cpp | cpp | Programa que recibe como argumento un entero "pos", y lo pasa como paramámetro al método pinDeDatos() de la clase PuertoParalelo.<br>El pin al que se accede es el número 4. Se activa o desactiva el pin si el número pos es mayor que 0 o no. | Incluye:<br>- PuertoParalelo.cpp<br><br>PINK_SWITCH |
| | testAt | ejecutable | Ejecutable de TestAtenuador.cpp | |
| | TestAtenuador.cpp | cpp | AteanuadorPga.atenua(0) | Incluye:<br>- AtenuadorPga.cpp |
| | testDeCalibracion.cpp | cpp | Define e implementa la clase ControlDeCalibracion. No se ha acabado, y tampoco se usa en ningún lugar. | Incluye:<br>- AtenuadorPga.h<br>- Sonometros.cpp<br><br>MAX_POINTS |
| | testPga | ejecutable | Ejecutable de testPga.cpp | |
| | TestPga.cpp | cpp | AtenuadorPga.atenua(at)<br><br>at es un entero que se pasa al programa por línea de comandos | Incluye:<br>- AtenuadorPga.cpp (y por cascada, incluyes todos los include de AtenuadorPga) |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | TP.cpp | cpp | Test PueroParalelo.cpp. Solo contiene un main al que se le pasa un entero por línea de comandos (x), para luego instancia un objecto de PuertoParalelo y activar solo el pin de datos x. El resto de pines se ponen a false (son 8 en total, un byte) | Incluye:<br>- AtenuadorPga.cpp (y por cascada, incluyes todos los include de AtenuadorPga) |
| **reports/** | | | | |
| | a.out | ejecutable | | |
| | cabeceraInformes.svg | SVG | | |
| | dataGet.cpp | | Versión "desactualizada" de getData | |
| | dygraph-combined.js | | https://dygraphs.com/ | |
| | excanvas.min.js | | https://github.com/GerHobbelt/excanvas | |
| | getConfig.cpp | | Genera informes sobre la configuración del limitador en varios formatos:<br>- JSON<br>- XML | Incluye:<br>- limitador/Registrador.cc<br>- limitador/Registro.cpp<br>- limitador/Estado.cpp<br>- limitador/Calibracion.cpp<br>- comun/Configuración.cpp |
| | getdata | ejecutable | | |
| | getData | ejecutable | | |
| | getData.cpp | | Genera informes sobre los registros en varios formatos:<br>- Uri ¿?<br>- JSON<br>- XML<br><br>Recibe 4 parámetros:<br>- formato, fechaIni, fechaFin, intervalo entre registros | Incluye:<br>- limitador/Registrador.cc<br>- limitador/Registro.cpp<br>- limitador/Estado.cpp<br>- comun/Configuración.cpp |
| | getEvents | script PHP | Script que recibe 3 parámetros, formato, fecha de inicio y fecha de fin, de modo que la invocación del script sería como sigue:<br><br>php?> getEvents <format> <startDate> <endDate><br><br>donde format puede ser **uri** o **json** y las fechas se dan en formato **year/month/day-hour:minute**<br><br>El script abre el fichero de logs y recupera los logs de eventos (aquellas acciones llevadas a cabo por usuarios) ocurridos entre las fechas de inicio y final, y los devuelve en el formato specificado. URI == raw -> los devuelve tal y como los lee del fichero de logs | Fichero de logs: /var/slr/logs.serial<br><br>Se hace referencia a este script en commv2/main.cpp |
| | getInputs.cpp | | Genera informes sobre las entradas del limitador. Los muestra por pantalla usando printf. | |
| | getStatus.cpp | | Genera informes sobre el estado del limitador en varios formatos:<br>- Uri ¿?<br>- JSON<br>- XML | Incluye:<br>- limitador/Lectura2C.cpp<br>- limitador/Calibracion.cpp |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | graphicalReport | directorio | Contiene una serie de ficheros con patrones de texto y un fichero CPP con el mismo nombre que el directorio.<br><br>No lo han completado y no se usa | |
| | gRegList.php | html, php, css, js | No se usa | |
| | gRep | directorio | Vacío | |
| | gReport.php | html, php, css, js | No se usa | |
| | Makefile | Makefile | Genera los ejecutables getStatus, getData, getConfig | |
| | testPerformance.php | php | Nada importante | |
| **scripts/** | | | | |
| Todos estos scripts se mandan a /bin | connectNetwork | script bash | Inicializa wvdial | wvdial |
| | continuousPink | script bash | Bucle infinito de “ruido rosa” | alsaplayer |
| | controlDeCalibracion | | Se apoya fuertemente en la librería de funciones calibrationControl.<br><br>Registra la actividad del equipo y genera la media de las lecturas para los últimos 10 minutos. | Incluye:<br>- www/calibractionControl.lib |
| | keepComm | script bash | Bucle infinito. Parece que comprueba cada cierto período de tiempo si hay conexión a internet haciendo un ping a Google,<br>en tal caso ejecuta slrComm, en otro caso regenera la conexión con connectNetwork | killComm |
| | keepLeds | script bash | Bucle infinito. Ejecuta ledControl cada segundo. | ledControl |
| | keepLm | script bash | Bucle infinito. Reinicia constantemente el limitador y registra el timestamp en var/slr/restarts | bin/limitador |
| | keepScreen | script bash | Mantiene la terminal activa para ledControl con screen /var/slr/leds/port en un bucle while infinito. | screen |
| | keepTime | script bash | Bucle infinito. Cada 60 segundos comprueba si hay conexión a internet realizando un ping a Google. En caso afirmativo, actualiza la hora con ntpdate y duerme durante 1000min(16h). Para actualizar la hora del sistema primero re-monta el sistema de archivos en modo lectura-escritura, actualiza la hora y luego lo devuelve a modo solo lectura. | |
| | killComm | script bash | Bucle infinito. Espera 4 HORAS antes de matar el proceso slrComm | |
| | ledControl | script PHP | Solo invocado por keepLeds. Se apoya en las funciones de las librería calibrationControl.lib y define e implementa algunas funciones propias. Abre una terminal con screen y muestra la media de las medidias del micrófono en ella, cambiando el color de la fuente según la cercanía de las medidas a los umbrales máximos (presión / dBs). Muestra el estado y la media de las medidas (mic, Ld, Li), caracter a caracter, usando echo con redirección a la terminal abierta por screen. | Incluye:<br>- www/calibractionControl.lib |
| | playPink | script PHP | Igual que continuousPink pero en PHP | |
| | refreshProcess | Script sh | Bucle infinito. Mata al proceso slrComm cada 15min (lo refresca) | |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | remoteShellService | script bash | Bucle infinito. [Función desconocida] Establece una conexión mediante netcat con boanergesnetwork.com , en el puerto $serial = {limInfo n}. Una vez establecida la conexión ejecuta /bin/localservice.<br><br>Se repite cada 30 segundos.<br><br>La web citada arriba parece ser una herramienta de acceso remoto. LDAP (Lighweight Directory Access Protocol) | nc boanergesnetwork.com -e bin\localservice |
| **tests/** | | | | |
| | a.out | ejecutable | | |
| | estadisticos.cpp | cpp | bubblesort y media arimética sobre un vector de float | |
| | estadisticos.php | php | Pruebas sobre getData. No se usa. | |
| | normativaExtendida.cpp | cpp | Tests sobre los valores máximos permitidos en los días semanales y los días fesitvos según la normativa extendida. (La cual desconozco) | incluye comun/Configuracion.cpp |
| | tActivo.cpp | cpp | Test que comprueba si el sistema está activo. Instancia un objeto de la clase Configuracion, y llama a los método carga() y estaActivo()<br><br>- carga(): carga el fichero de configuración "/var/slr/configuracion" (definido en comun/Config.h) y desencripta su contenido.<br><br>**NOTA**<br>Al parecer han definido un encriptado a nivel de bits para almacenar la información sensible de la configuración del sistema, como los detalles del local, claves y demás.<br>Cada uno de estos campos se almacena en un vector de tipo char. El vetor se rellena caracter a caracter coo resultado del desencriptado. El desencriptado consta de la suma de dos caracteres que dependen del caracter encriptado x:<br>- p1 = (x & 0x0F)<<4<br>- p2 = (x & 0xf0)>>4<br>- x = p1 + p2 | |
| | testDeCal.cpp | cpp | Realiza un test de calibración. Define y configura un objeto de la clase Calibracion.<br><br>Hace una llamada al método calcula_dbs de dicha clase | incluye limitador/Calibración.cpp |
| | tReg.cpp | cpp | Llama al método obtenPosicion(time) de la clase Registrador. Este método devuelve un número entero que corresponde al nº de registro de la hora dada. | incluye limitador/Registrador.cpp |
| **utiles/** | | | | |
| | a.out | ejecutable | | |
| | autoEq | ejecutable | | |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | AutoEqualizador.cpp | cpp | Función getLecturas y main.<br><br>getLecturas(int lecturas, Lectura2C &mic, Lectura2C &lin)<br>- Lee el número de lecturas indicado desde los archivos "/tmp/.lx[0-1]", donde 0 es la lectura del micrófono y 1 la lectura de las líneas en ese momento. Las lecturas se van almacenando en dos vectores para luego obtener sus medias y devolverlas en mic y lin.<br><br>El main se tiene Lectura2C media_mic, media_lin y Calibracion media, izquierda, derecha. Se leen las medias y las calibraciones de cada línea (previamente se calibran), se calcula la ecualización para cada línea y se aplican. Finalmente se escribe a fichero las calibraciones. | Incluye:<br>-limitador/Lectura2C.cpp<br><br>NumeroDeLecturas |
| | BoanergesConfigurator.cpp | cpp | Contiene una función print y un main. El print muestra por pantalla algunos datos de configuración. El main permite recibir de forma opcinal 2 parámetros. De forma general, la invocación sería ./BoanergesConfigurator <Atributo> <Valor>. Los valores admitidos para *Atributo* son: [Server, ServerDataRevission, ServerPort, GuidRemoto, ServerKey, LocalKey]. Cuando se recibe uno de estos atributos, se actualiza el valor de dicho atributo en la configuración al valor pasado como segundo argumento, tanto en memoria como en disco. Solo se permite actualizar un atributo cada vez. Si no se reciben argumentos, se muestra la configuración actual por pantalla. | Incluye:<br>- comun/Configuracion.cpp |
| | controlDeCalibracion | script php | Controla el nivel de calibrado del sistema.<br><br>TODO usa muchas funciones de la librería, las cuales habría que estudiar | Incluye:<br>- www/calibractionControl.lib |
| | gen | ejecutable | | |
| | gen.cpp | cpp | Generador "aleatorio" de cadenas. ¿Para contraseñas? | |
| | getSerial.cpp | cpp | Muestra por pantalla el número de serial (ID) del dispositivo, y lo vuelca en serial.h | Incluye:<br><br>- comun/serial.h |
| | gSerial | ejecutable | | |
| | helper.cpp | cpp | No se usa. Muestra por pantalla información general sobre el limitador. Es menos completo que LimInfo | Incluye:<br><br>- comun/Configuracion.cpp<br>- comun/EstadoDelLimitador.cpp |
| | Inicializador.cpp | cpp | Arranca e inicializa el registrador | Incluye:<br><br>- limitador/Registrador.cc |
| | LimInfo.cpp | cpp | Imprime por pantalla ifnromación relativa al limitador, estado, configuración, detalles del local, etc. El programa recibe como parámetros una cadena de caracteres los cuales indican qué mostrar, por ejemplo ./limInfo 5 muestra la dirección del local,./limInfo A muestra el valor de atenuación máxima, y . /limInfo A5 muestra ambas cosas | Incluye:<br><br>- comun/Configuracion.cpp<br>- comun/EstadoDelLimitador.cpp<br>- comun/Funciones.h<br>- comun/Serial.h |
| | Linet.cpp | cpp | Obsoleto. Utiliza una clase llamada Comunicador, la cual no existe. La busca dentro de la carpeta comun/ | |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | Makefile | Makefile | Genera los siguientes archivos:<br>- utilslr<br>- gen<br>- limInfo<br>- localservice<br>- inicializador<br>- boanerges.config<br>- eq<br>- gSerial | Ejecuta y redirecciona la salida de gSerial (getSerial.cpp) a serial.h |
| | playPink | script php | Lanza alsaplayer para que reproduza ruida rosa en un bucle infinito. | alsaplayer -d hw:1,0 /var/slr/pink.wav<br>pinkVolume: amixer -c 1 sset PCM $nivel<br><br>Incluye:<br>- www/calibractionControl.lib |
| | serial.h | h | Archivo vacío. Recibe la salida de gSerial (getSerial.cpp) | |
| | ServidorSerie.cpp | cpp | Compilado como localservice, tiene como finalidad el crear una especie de interprete de comandos / consola remota, la cual acepta las siguientes órdenes:<br>- datos:<br>- update:<br>- configuracion:<br>- configura:<br>- activa:<br>- concha:<br>- concharemota:<br>- identifica:<br>- calibra:<br>- prepara:<br>- operativo:<br>- test:<br>- calibracion:<br>- salir:<br>- ayuda:<br>- sql:<br>- xml:<br>- yml:<br>- id:<br>- version:<br>- identificame:<br>- identificate:<br><br>- calibralineas:<br>- mic:<br>- miceq:<br>- lefteq:<br>- righteq: | Incluye:<br>- serial.h<br>- comun/aes.cpp<br>- comun/Configuracion.h<br>- comun/Configurador.cpp<br>- comun/EstadoDelLimitador.h<br>- comun/Funciones.cpp<br>- limitador/Lectura2C.cpp<br>- limitador/Registrador.cc<br>- limitador/AtenuadorPga.cpp<br><br>Incluye condicionalmente:<br>- comunicacion/servidor.cpp<br><br>La carpeta que contiene este archivo esta marcada como old, por lo que imagino que este interprete quedó obsoleto. |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| **Importante** | utilslr.cpp | cpp | Backend de la interfaz web. La interfaz web en php se comunica con este programa de forma extensiva, por ejemplo para el login, se invoca a este programa de la siguiente forma:<br>php $> utilslr clave <password> | Incluye:<br><br>- comun/Funciones.h<br>- comun/Configuracion.cpp<br>- comun/Configurador.cpp |
| | x.cpp | cpp | Lee y muestra por pantalla el último registro del sistema, representado por la macro F_Registro | incluye:<br>- limitador/Registro.cpp<br>- limitador/Registrador.cc |
| **www/** | | | | |
| | access.php | php | Conjunto de funciones para la gestión de usuarios y login. Usa el fichero users.auth como base de datos | /var/slr/users.auth |
| | authPart-ExtendedNormative.php | php, html, css, js | Para la gestión de la normativa extendida (máximos permitidos por día y hora)<br><br>Petición AJAX a functions/updateExtendedNormative.php | Usa getConfig, limInfo |
| | calibrationControl.lib | php | Colección de funciones. Se entienden fácilemente y están brevemente comentadas. Tal y como el nombre sugiere son funciones relacionadas con el control de calibración y las sesiones de ruido. Contiene funciones para comenzar y parar a emitir ruido rosa, comenzar y parar sesiones de calibrado, aplicar y eliminar atenuación a las líneas, obtener información del limitador (delegando en getStats, getConfig..) y leer / escribir logs (delegando en logs.php) | Incluye:<br>- logs.php |
| | configFail.inc | html | Trozo de HTML a mostrar cuando la identificación falla | |
| | configPart.php | | Vacío | |
| | configuration.php | php | Define lo siguiente:<br><br>$GlobalConfiguration[reportOutputFile]="/tmp/reportEngine.output";<br>$GlobalConfiguration[reportsBaseDir]="/var/slr/tpl/";<br>$GlobalConfiguration[reportOutputType]="/tmp/reportEngine.output.type"; | |
| | config.xml | xml | Información del equipo y valores por defecto | |
| | css | css | -- | |
| | fonts | | -- | |
| **Importante** | functions | php | Funciones AJAX. Comunicación con el backend. | functions/calibrate.php:<br><br>Utiliza localservice para aplicar los cambios |
| | help.png | | -- | |
| | images | | -- | |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | index.php | php, html, js, css | Controlador principal de la página web. Contiene algunas funciones PHP, principalmente para la gestión del login y la reconstrucción a partir de la sesión.<br><br>PHP:<br>La página cambia dependiendo de si el equipo es tan sólo un registrador o si también tiene capacidades de limitación acústica. Por ello, se comprueba al acceder a la página web en qué modo se encuentra el equipo, consultando esta información mediante `limInfo A` y se guarda el resultado en la variable $isARegister<br><br>A continuación se comprueba si se ha iniciado sesión o si se está intentando acceder. Hay varios modos de acceder al sistema:<br>- Usando DNI y contraseña<br>- Usando solo contraseña: usa `utilslr clave`<br>- Entrando en modo consultor<br>En cualquier caso, actualiza la variable user en consecuencia y escribe logs de las acciones tomadas, y se mantiene el estado del fichero /tmp/loguedOn (se crea o se destruye)<br><br>Se construye el HTML de manera dinámica con PHP. En el head se inyectan las librerías JS y en el body se incluyen unos scripts PHP u otros dependiendo de varios factores, como si es registrador, si se está logueado o si es la primera vez que se incia el equipo. | $_SESSION['user'] = $user;<br>$_SESSION['logueado'] = true;<br><br>Donde $user es un array devuelto por la función getUser de access.php, que contiene el siguiente formato cuando el login ha tenido éxito (mismo formato que en el fichero /var/slr/users.auth):<br><br>$user = array(<br>"found" => 1,<br>"dni" => $dni,<br>"name" => $name,<br>"userManager" => $userManager,<br>"configManager" => $configManager,<br>"password" => $password,<br>"updateTime" => $time<br>); |
| | jquery.css | css | -- | |
| | js | | Librerías JS de terceros como jQuery | |
| | leftEqualizer.php | | Web UI -> Calibrar -> Ecualizador izquierdo<br><br>Petición AJAX a functions/calibrate.php | |
| | login.php | php | Script que controla el login. Ejecuta utilslr clave <password> | |
| | logoPagina.png | | | |
| | logs.php | php | Escribe logs en estos dos ficheros:<br>- /var/slr/logs.serial -> LogsFile<br>- /var/slr/sessions.serial -> SessionsFile<br><br>Escribe todo lo relativo a usuarios (login, logout...) y lo relativo a sesiones (comienzo y fin de sesión y hora de arranque del sistema). Lo primero va a logs, lo segundo a sessions. Los logs de los sesiones los guarda en base64, mientras que los de usuario los almacena en texto plano. | |
| | manual3.pdf | pdf | Manual para el registrador de sonido | |
| | manual7.pdf | pdf | Manual para el limitador de sonido | |
| | manual.php | php | Envía el manual para su descarga. | |
| | new | | Solo contiene un fichero, tabAuth.php, el cual parece ser una mejora del tabAuth existente | |
| | old | | Código descartado / antiguo / no usado | |
| | reportOutputTest.html | | -- | |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | reports | | **TODO** | |
| | rightEqualizer.php | | Web UI -> Calibrar -> Ecualizador derecho<br><br>Petición AJAX a functions/calibrate.php | |
| | scripts.js | | Realiza una petición AJAX a functions/statys.php para obtener el estado del limitador. Tras obtener los datos, los procesa y actualiza la IU (la parte de statusPart.php) | |
| | simple | | Web simple que muestra la parte central de la web normal (barras y gráfico) en formato reducido | |
| | smallInfoPanel.php | | **No se usa.** Corresponde al panel de información en el menú desplegable, el que contiene los sliders para configurar el equipo. | |
| | statusPart.php | | Parte del estado actual del limitador. La parte central de la UI que muestra las lecturas actuales.<br><br>Peticón AJAX a functions/status.php mediante scripts.js | Incluye:<br>- scripts.js |
| | style-Blue.css | css | -- | |
| | style.css | css | -- | |
| | tabAuth.php | | Menú desplegable superior cuando se está identificado.<br><br>Peticiones AJAX redirigidas a los siguientes scripts en la carpeta functions/:<br>- updatePwd<br>- updateCnfg<br>- calibrate<br>- pink<br>- updateDataCu<br>- validateUserCode<br>- network<br>- updateTime | |
| | tabAuthRegister.php | | No se usa (comentado). Menú desplegable superior cuando se está identificado y el equipo es un registrador.<br><br>Peticiones AJAX redirigidas a los siguientes scripts en la carpeta functions/:<br>- updatePwd<br>- soundControl<br>- calibrate<br>- updateDataCu<br>- validateUserCode<br>- network | |

| Directorio | Archivo | Tipo | Descripción | Notas de interés |
|---|---|---|---|---|
| | tabNormal.php | | Menú desplegable superior cuando NO se está identificado.<br><br>Peticiones AJAX redirigidas a los siguientes scripts en la carpeta functions/:<br>- updatePwd<br>- soundControl<br>- calibrate<br>- updateDataCu<br>- validateUserCode<br>- network | |
| | templateMng.php | | No se usa. | |
| | tests | | -- | |
| | tractis | | http://www.tractis.com/ | |
| | updateDataCu.php | | Actualiza los datos del local. Para ello escribe la nueva información en el archivo /tmp/conf.tmp y luego ejecuta `utilslr configura /tmp/conf.tmp` | |
| | users.php | | Gestión de usuarios | |