

## TRABAJO FINAL DE GRADO

### Grado en Ingeniería de Sistemas Audiovisuales

# ESTUDIO Y DESARROLLO DE UN INTERFAZ PARA LA OBTENCIÓN DE LA SEÑAL DE TALLY PROVENIENTE DE UN MEZCLADOR DE VÍDEO



## Memoria i Anexos

Junio 2019

**Autor:** Víctor Pacheco García  
**Director:** Juan Mon González  
**Co-Director:** Juanjo Alins Delgado



## Resumen

En el presente trabajo se ha implementado un **sistema que permite obtener la señal de tally de un mezclador de vídeo ATEM de Blackmagic Design y mostrarla en tiempo real en forma de indicador luminoso**. Dicha señal va a permitir discernir cuál es la fuente que se encuentra en anticipo y cuál en programa, pudiendo indicar al operador de cámara si la señal que está captando está preparada para emitirse en directo o se está emitiendo. Esta información es crítica durante una realización audiovisual y puede gestionarse también mediante dispositivos de intercomunicación de audio. Sin embargo, estos últimos se utilizan para la coordinación entre todos los equipos técnicos (sonido, iluminación, regidor, cámaras, etc.), por lo que no es conveniente congestionar este canal de comunicación.

Existen múltiples sistemas en el mercado con dicho propósito que se analizan en el Estado del Arte (véase página 3). Todos ellos constan de un dispositivo que se conecta al mezclador mediante un cable Ethernet, extraen la información de *tally* del bus de datos y en base a esta, actúan sobre los indicadores luminosos de los dispositivos receptores ubicados en las cámaras.

Tanto el software como el hardware del sistema realizado están basados en **código abierto**, es funcional y utiliza tecnología inalámbrica que facilita el montaje y la movilidad. Además el sistema es adaptativo, ya que selecciona el canal Wi-Fi con menos interferencia, permite adaptar la luminosidad de los indicadores en función de las condiciones lumínicas del entorno, así como asociar los dispositivos receptores con un número de cámara sin necesidad de modificar el código.

Para este propósito, se han estudiado los comandos del protocolo de red utilizado por el mezclador ATEM y las librerías *Arduino* de código abierto que proveen de las funciones para conectar y controlar los mezcladores de vídeo ATEM, obtenidos por la empresa *SKAARHOJ* a través de ingeniería inversa.

En base dichos recursos, se ha utilizado el software **Arduino IDE** (*Integrated Development Environment*), un entorno de desarrollo integrado multiplataforma cuyo lenguaje está basado en *wiring*, que a su vez lo está en *Processing* y cuyas librerías están escritas en C/C++. En cuanto a hardware, se han utilizado placas de desarrollo con un *SoC* (System on chip) Wi-Fi **ESP8266** de Espressif Systems y un módulo ESP-12.

Se han realizado los scripts necesarios para obtener la señal de *tally* del mezclador y distribuirla a otros dispositivos mediante el estándar 802.11 b/g/n de redes **Wi-Fi** que opera en la banda de 2.4 GHz. Así mismo, se ha diseñado una placa de circuito impreso mediante el software de diseño electrónico KiCad para integrar y conexionar los componentes que conforman el receptor final de la señal de *tally*. Este cuenta con dos indicadores led, un potenciómetro para regular la intensidad luminosa de estos, un interruptor eléctrico DIP (del inglés: *Dual In Line Package*) para determinar a qué número de cámara corresponde y un duplicador USB que se utiliza para proporcionar la alimentación necesaria al receptor. Además, el dispositivo dispone de un adaptador para acoplarse a cualquier cámara con zapata para *flash*.

## Abstract

The aim of this project has been to implement a **system to obtain and transmit the tally signal of a Blackmagic Design's ATEM mixer and display it in real time as a light indicator**. The tally signal allows discerning which source is on the program output and which one is on preview. In this way, the camera operator knows if the signal of his camera is ready to be broadcasted live or is already on air. This information is critical during an audiovisual production and can also be managed by audio intercommunication devices. However, the latter are used for coordination between all the technical teams (sound, lighting, alderman, cameras, etc.), so it is not convenient to congest this communication channel.

There are multiple systems in the market for this purpose that are analyzed in the State of the Art (see page 3). All of them consist of a device that is connected to the mixer by an Ethernet cable. It extracts the tally information from the data bus and, based on it, they act on the luminous indicators of the receiver devices located in the cameras.

The system is **open-source** based, it is functional and uses wireless technology that facilitates the installation and the mobility. The system is also adaptive: it can automatically select the WIFI channel with less interference, it permits to customize the intensity of the light indicators and it can also associate the receiver devices with a camera number, without modifying of the code.

For this purpose, it has studied the network protocol used by the ATEM mixer and the open source Arduino libraries that provides the functions to connect and control the ATEM mixers, obtained by the company SKAARHOJ through reverse engineering.

Based on these resources, it has used the **Arduino IDE** (*Integrated Development Environment*) software, a multiplatform integrated development environment whose language is based on wiring, which in turn, is build on Processing and whose libraries are written in C / C ++. In terms of hardware, development boards have been used with a Wi-Fi **ESP8266** SoC (System on chip) from Espressif Systems and an ESP-12 module.

The necessary scripts have been made to obtain the tally signal of the mixer and delivers it to other devices through the 802.11 b /g /n standard of **Wi-Fi** networks operating in the 2.4 GHz band. Likewise, a printed circuit board has been designed using the electronic design software KiCad, to integrate and connect the components that make up the final receiver of the tally signal. The device has two LED indicators, a potentiometer to set the luminous intensity of the LED, an electrical switch DIP (Dual In Line Package) to determine which camera number is associated with it and a USB duplicator that is used to provide the necessary power to the receiver. In addition, the device has an adapter that permits to attach the device to any camera with hot shoe.

## Agradecimientos

A Juan Mon y Juanjo Alins. Por su tiempo, conocimientos, dedicación y entrega. Por hacer posible el proyecto y por hacer de este una experiencia enriquecedora y estimulante. A Dani Pérez, por su colaboración desinteresada e inmensa paciencia.

A mis padres, por estar siempre ahí, por apoyar mis decisiones y empujarme a alcanzar mis metas.

Per Valentina, per esserci ogni giorno, perché con te è tutto più facile. Per il tuo amore incondizionato.

A mis amigos y amigas, por ser fuente de inspiración, por que no puedo más que sentirme afortunado de compartir mi vida con todos vosotros.

A SKAARHOJ ApS, por compartir el fruto de su trabajo.

*“La gratitud es una cualidad similar a la electricidad: debe producirse, descargarse y agotarse para poder existir” (William Faulkner).*

# Índex

<b>RESUMEN</b>	<b>I</b>
<b>ABSTRACT</b>	<b>II</b>
<b>AGRADECIMIENTOS</b>	<b>III</b>
<b>DECLARACIÓN DE HONOR</b>	<b>VIII</b>
<b>1. INTRODUCCIÓN</b>	<b>1</b>
1.1. Prefacio .....	1
1.2. Objetivos y Alcance.....	1
<b>2. ESTADO DEL ARTE</b>	<b>3</b>
2.1. Revisión del estado del arte .....	3
2.2. Crítica al estado del arte .....	7
<b>3. PROPUESTA</b>	<b>8</b>
3.1. Decisión sobre soluciones alternativas .....	9
3.1.1. Medio de transporte de la señal de <i>tally</i> .....	9
3.1.2. Placas de desarrollo .....	11
<b>4. DESARROLLO SOFTWARE</b>	<b>13</b>
4.1. Estudio de las librerías de SKAARHOJ.....	13
4.1.1. Constructor objeto de la clase ATEM. ....	13
4.1.2. Función <i>setup_ethernet</i> de la clase ATEM <i>Tally</i> .....	16
4.2. Desarrollo software de la estación base .....	17
4.2.1. Implementación de las librerías SKAARHOJ con Arduino IDE. ....	18
4.2.2. Implementación de la conectividad Wi-Fi.....	20
4.2.3. Envío de la información de <i>tally</i> por UDP multicast.....	22
4.2.4. Selección del canal Wi-Fi.....	23
4.3. Desarrollo software de las estaciones receptoras.....	25
4.3.1. Establecimiento de una estación Wi-Fi .....	26
4.3.2. Asociación a un número de cámara .....	26
4.3.3. Recepción de la información de <i>tally</i> por UDP multicast y gestión de los ledes .	26
4.3.4. Regulación de la intensidad luminosa de un diodo electroluminiscente .....	27
<b>5. DESARROLLO HARDWARE</b>	<b>29</b>
5.1. Desarrollo hardware de la estación base.....	29

5.2.	Desarrollo hardware de las estaciones receptoras .....	29
5.2.1.	Desarrollo de la placa de circuito impreso (PCB) .....	31
5.3.	Diseño hardware de un punto de acceso externo .....	34
<b>6.</b>	<b>DISEÑO DEL PROTOTIPO DE LAS ESTACIONES RECEPTORAS</b> .....	<b>35</b>
<b>7.</b>	<b>CONCLUSIONES Y TRABAJOS FUTUROS</b> .....	<b>37</b>
<b>8.</b>	<b>PRESUPUESTO</b> .....	<b>39</b>
8.1.	Presupuesto de la estación base .....	39
8.2.	Presupuesto de una unidad de estación receptora .....	39
8.3.	Presupuesto del punto de acceso externo .....	39
<b>9.</b>	<b>BIBLIOGRAFÍA</b> .....	<b>41</b>
<b>ANEXOS</b>	<b>.....</b>	<b>43</b>
ANEXO 1.	Comparativa del Estado del Arte .....	43
ANEXO 2.	Especificaciones Wi-Fi del ESP8266 .....	44
ANEXO 3.	<i>Sketch</i> de las estaciones receptoras .....	45
ANEXO 4.	<i>Sketch</i> de la estación base sin punto de acceso propio .....	47
ANEXO 5.	<i>Sketch</i> del punto de acceso .....	49
ANEXO 6.	<i>Sketch</i> de la estación base con punto de acceso propio .....	50
ANEXO 7.	Hoja de Características del BSS138 .....	53
ANEXO 8.	Hoja de características de la caja TWN 5-4-7 .....	55
ANEXO 9.	Diagrama de Gant .....	56

## Índice de ilustraciones

<b>Ilustración 1.</b> Esquema técnico para embeber la señal de <i>tally</i> con la señal de audio _____	3
<b>Ilustración 2.</b> GPI and Tally de Blackmagic (izquierda) y conector DB25 (derecha) _____	4
<b>Ilustración 3.</b> Intercom ITC-100 (izquierda inferior), lámpara TD-2 (izquierda superior) y petaca ITC-100SL (derecha) _____	4
<b>Ilustración 4.</b> Estación y lámpara de CEREVO _____	6
<b>Ilustración 5.</b> Receptor Cuebi _____	6
<b>Ilustración 6.</b> Esquema técnico propuesto _____	9
<b>Ilustración 7.</b> Diagrama de bloques de la estación base con punto de acceso propio _____	17
<b>Ilustración 8.</b> Captura del tráfico a través de la red WiFi formada con el punto de acceso creado por el ESP8266, con filtro por protocolo UDP, fuente y destino (izquierda). Gráfico de la tasa de paquetes UDP capturados según los filtros indicados (derecha) _____	20
<b>Ilustración 9.</b> Señal IEEE 802.11 medida utilizando el analizador de espectro Keysight _____	23
<b>Ilustración 10.</b> Diagrama de bloques de las estaciones receptoras _____	25
<b>Ilustración 11.</b> Tabla de conexión entre el conector ICSP del módulo Ethernet y la placa WEMOS (izquierda). Cableado realizado (derecha) _____	29
<b>Ilustración 12.</b> WEMOS D1 R1 con módulo Ethernet y caja DesignSpark _____	29
<b>Ilustración 13.</b> (Izquierda) BSS138 Datasheet. Fairchild Semiconductor, Octubre 2005. (Centro) Puerta lógica NOT, con un N-MOSFET. (Derecha) Análisis transitorio del circuito tipo realizado con Tina V9 _____	30
<b>Ilustración 14.</b> Esquemático (izquierda) y <i>layout</i> realizados con KiCad 5.0.2 _____	32
<b>Ilustración 15.</b> Vista 3D de la PCB generada con KiCad 5.0.2 _____	33
<b>Ilustración 16.</b> Vista de los archivos Gerber de las capas trasera (izquierda) y frontal (derecha) realizados con KiCad 5.0.2 _____	33
<b>Ilustración 17.</b> PCB realizada con la fresadora ProtoMat S62 (izquierda). Soldado de componentes a la PCB (derecha) _____	34
<b>Ilustración 18.</b> Diseño 3D del dispositivo receptor realizado con Maya 2018. Planteamiento inicial _____	35
<b>Ilustración 19.</b> Diseño 3D final del dispositivo receptor realizado con Maya 2018 _____	36
<b>Ilustración 20.</b> Vista cenital del prototipo (izquierda), vista frontal (centro) y vista lateral (derecha) _____	36
<b>Ilustración 21.</b> Especificaciones técnicas del Wi-Fi del ESP8266 _____	44



## Índice de Abreviaturas

**Arduino IDE** (del inglés: Arduino Integrated Development Environment)

**CNC** (del inglés: *Computer Numeric Control*)

**CPU**(del inglés *Central Processing Unit*)

**EDA** (del inglés: *Electronic Design Automation*)

**GPIO** (del inglés: *General Porpouse Input/Output*)

**GPL** (del inglés: *General Public Liscence*)

**I/O** (del inglés: *Input/Output*)

**ICSP** (del inglés: *In-Circuit Serial Programming*)

**IP** (del inglés: *Internet Protocol*)

**MOSFET** (del inglés: *Metal Oxide Semiconductor Field Effect Transistor*)

**NC** (normalmente cerrado)

**P.V.P** (Precio de Venta al Público)

**PWM** (del inglés: *Pulse Width Modulation*)

**RFC 2365** (del inglés: *Request For Comments*)

**SDI** (del inglés: *Serial Digital Interface*)

**SM** (del inglés: *Industrial, Scientific and Medical*)

**SMPTE** (del inglés: *Society of Motion Picture and Television Engineers*)

**SoC** (del inglés: *System on Chip*)

**SPI** (del inglés: *Serial Peripheral Interface*)

**SSID** (del inglés: *Service Set Identifier*)

**TCP** (del inglés: *Transmission Control Protocol*)

**UDP** (del inglés: *User Datagram Protocol*)

**WLAN** (del inglés: *Wireless Local Area Network*)

# 1. Introducción

## 1.1. Prefacio

Un mezclador de vídeo es un interfaz que permite seleccionar, mezclar y manipular las fuentes de vídeo que en este se conectan. Los mezcladores ATEM, permiten procesar diversas fuentes multimedia (no solamente vídeo) y alternar entre estas durante la realización de un evento. Estos dispositivos están operados por el realizador, que será el responsable de seleccionar la fuente que va a ser emitida o grabada en cada momento, decidiendo así qué, cómo y cuándo se verá un contenido audiovisual. Por ello, es importante que el operador de cámara sepa, en cada momento, si la señal de vídeo captada por su cámara está siendo emitida (señal de programa) o está seleccionada para emitirse (señal de anticipo), para actuar en consecuencia.

Tras la utilización del mezclador ATEM en las asignaturas de "Equips de video" y "Producció Audiovisual" del grado en Ingeniería de Sistemas Audiovisuales (UPC-ESEIAAT), se detectó la necesidad de disponer de un sistema de *tally*. Este, junto con los intercomunicadores de audio, son vitales para la realización de eventos en directo, puesto que permiten la comunicación entre los operadores de cámara y el equipo de realización. Un ambiente ruidoso como el de un concierto, interferencias en los equipos de intercomunicación, la necesidad de ajustar la ganancia de los auriculares al mínimo para no interferir en el evento a realizar y otras problemáticas, pueden hacer que los operadores no sean capaces de discernir con claridad las órdenes que reciben a través de los intercomunicadores. Estas y otras casuísticas, comprometen la correcta realización de video, puesto que los operadores pierden la referencia de la señal que está siendo emitida. Por tanto, la señal de *tally* no solo es un complemento de los *intercom*, sino que en ocasiones es la única forma de comunicación de los realizadores con los operadores de cámara.

El hallazgo del trabajo de *SKAARHOJ* abrió la posibilidad de realizar un producto real y funcional en tan solo un cuatrimestre, dándonos la motivación para enfrentarnos a este reto. El presente proyecto aborda el mecanismo necesario para que el realizador informe a cada operador de cámara del estado de la señal captada por su cámara.

## 1.2. Objetivos y Alcance

Se quiere implementar una interfaz software y hardware que permita conectarse con un mezclador ATEM y obtener la información de *tally*, que indica cuales son las entradas de este que están en anticipo y programa. Esta información, deberá comunicarse a los interfaces que se ubicarán en cada una de las

cámaras conectadas al mezclador y que dispondrán de un sistema de indicadores luminosos para dar respuesta a dicha comunicación. Este sistema deberá cumplir las siguientes premisas:

- Todo el trabajo se elaborará con software y hardware libre.
- El sistema resultante deberá ser funcional.
- El tiempo de respuesta, desde que se da un cambio en la señal de anticipo o programa hasta que los ledes actúan en consecuencia, no deberá comprometer la realización.
- El sistema deberá tener un alcance mínimo, para que pueda ser utilizado en el plató del TR1 de la ESEIAAT.
- Los indicadores luminosos podrán regularse, para adaptar estos a las condiciones de luminosidad del entorno durante la grabación.
- Deberá ser compatible con la versión instalada del software Blackmagic ATEM Switchers 7.5.

Se pretende realizar un sistema funcional, que pueda utilizarse en las diferentes asignaturas del Grado en Ingeniería de Sistemas Audiovisuales de la UPC, mejorando ostensiblemente la comunicación del equipo de realización con los operadores de cámaras. Existe una limitación económica que restringirá las alternativas de los componentes hardware, en especial los que no son críticos para el funcionamiento del sistema.

No se pretende realizar un producto comercial, para su posterior venta. Por ello, no se profundizará en aspectos económicos ni se diseñarán los dispositivos con esa intencionalidad.

Se trata de un proyecto multidisciplinar que aglutinará el conocimiento adquirido en diferentes materias cursadas durante el grado, como *Circuits, Electrònica analògica, Sistemes i aplicacions telemàtiques, Equips de video, Continguts Multimèdia, Processadors digitals, Fonaments d'informàtica* y *Projectes d'enginyeria* entre otras.

## 2. Estado del arte

Los mezcladores de video ATEM del fabricante Blackmagic pueden comunicarse con cámaras y monitores mediante el envío de señales que identifican la fuente que está siendo transmitida. Esta señal utiliza un protocolo de señalización que, en parte, está disponible en el manual de estos productos [1]. Este protocolo es utilizado por otras empresas para diseñar interfaces para obtener y recodificar dicha señal, dando lugar a sistemas de *tally* alternativos a los que el propio Blackmagic ofrece.

A continuación se exponen los sistemas de *tally* más relevantes disponibles en el mercado. En el ANEXO 1 se adjunta una tabla a modo de resumen, que compara e indica el P.V.P (Precio de Venta al Público) de las diferentes alternativas expuestas, para un sistema que dé soporte a cuatro cámaras.

### 2.1. Revisión del estado del arte

#### 2.1.1. BLACKMAGIC

La primera solución de Blackmagic Design pasa por embeber las señales de intercom, de control de cámara y de *tally* en el flujo de datos de la señal de programa SDI (del inglés: *Serial Digital Interface*). Esta operación es llevada a cabo por el conversor ATEM Talkback Converter, que distribuye las señales citadas a las cámaras mediante 12G-SDI o fibra óptica SMPTE de manera opcional. Se requiere un monitor Blackmagic como el URSA Studio Viewfinder o una cámara como la Blackmagic Studio Camera que disponen de las entradas correspondientes e integran un indicador luminoso.

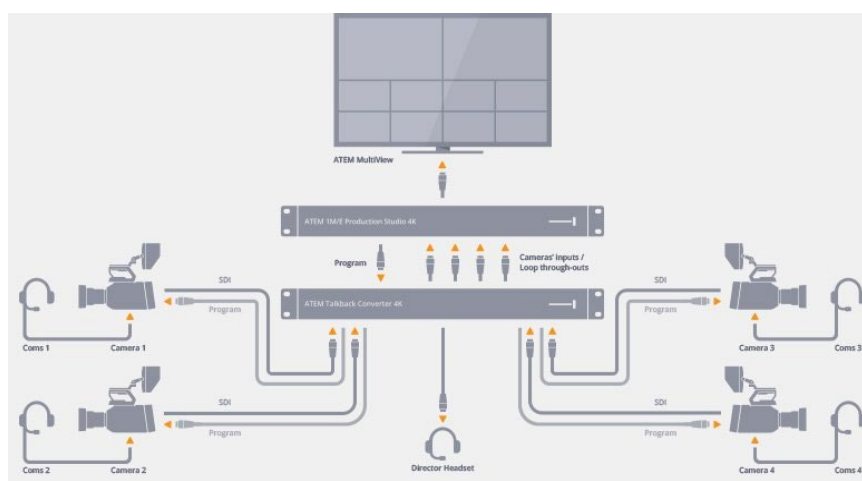


Ilustración 1. Esquema técnico para embeber la señal de *tally* con la señal de audio.  
Fuente: <https://www.blackmagicdesign.com>, Abril 2019.

Una segunda solución de Blackmagic es la interfaz GPI and Tally, que se conecta al mezclador vía Ethernet y proporciona 8 salidas mediante relés mecánicos con contactos NC (normalmente cerrado), conectados a tierra y con una tensión máxima de 30V y una corriente máxima de 1A [1]. Utiliza un conector tipo D para albergar estas salidas junto a 8 entradas en forma de interruptor óptico. Para llevar las señales proporcionadas por dicho conector hacia la cámara, Blackmagic no ofrece ningún producto, por lo que deben valorarse multitud de alternativas de conectores DB25 y lámparas.

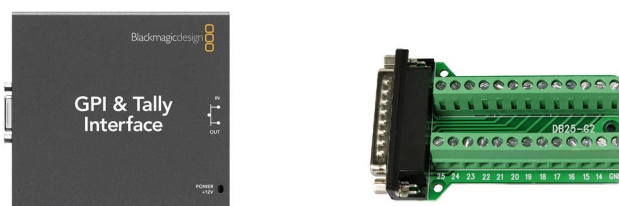


Ilustración 2. GPI and Tally de Blackmagic (izquierda) y conector DB25 (derecha). Fuente: <https://www.blackmagicdesign.com>, Marzo 2019.

### 2.1.2. DATAVIDEO

Una de las posibles soluciones para llevar la señales de salida del interfaz GPI and Tally hacia las cámaras es embebiéndola con la señal de audio del intercom ITC-100 de Datavideo. Dicha señal será transmitida mediante un cable XLR hacia las petacas ITC-100SL que disponen de un led a modo de indicador *tally* o bien extender dicha señal mediante la salida mini-jack del dispositivo hacia la lámpara TD-2 o TD-3.



Ilustración 3. Intercom ITC-100 (izquierda inferior), lámpara TD-2 (izquierda superior) y petaca ITC-100SL (derecha). Fuente: [datavideocorporation.desk.com/](http://datavideocorporation.desk.com/), Marzo 2019.

Por otro lado, Datavideo ofrece el conversor TB-5 que obtiene la señal de la interfaz *GPI & Tally* a través de un cable D-SUB25 macho a D-SUB15 macho y distribuye la señal de *tally* por medio de cuatro salidas jack de 3.5mm hacia cuatro receptores con indicadores luminosos TB-2/3.

### 2.1.3. SKAARHOJ

SKAARHOJ es una empresa de ingeniería y multimedia que diseña soluciones de control para sistemas broadcast. Parte de su hardware y software es lanzado bajo licencias de código abierto.

Tally Box System se conecta con el mezclador a través de Ethernet y proporciona 8 salidas Ethernet que corresponden a las 8 entradas que puede manejar el mezclador de manera simultánea. Las lámparas disponen de dos indicadores luminosos, uno en la parte trasera para el operador de cámara y otro en el frontal para el presentador. Las lámparas tienen una entrada y una salida Ethernet, siendo estas últimas para realizar una configuración en cadena. De este modo, con una sola salida del Tally Box, pueden suministrarse hasta tres señales de *tally* a tres cámaras diferentes.

En el contexto de nuestro trabajo, también ofrece diferentes conversores para mover señales sobre Ethernet, Wi-Fi, SDI y GPIO (del inglés: *General Porpouse Input/Output*). Un ejemplo es el ETH-GPI Link, basado en el Interfaz GPI/Tally de Blackmagic.

Sin embargo, el aporte diferencial de esta empresa es el material que ofrece de manera gratuita, del que destacaremos el protocolo de red usado por el controlador ATEM obtenido por ingeniería inversa y las librerías en C++, Arduino compatibles, bajo licencia GPL (General Public Liscence) asociadas a dicho protocolo. Estas librerías serán el punto de partida de nuestro proyecto [2].

### 2.1.4. KVITKO

Basado en las librerías de SKAARHOJ, Kvitko ha desarrollado un sistema de *tally* de código abierto, que consta de un dispositivo que se conecta a la salida Ethernet del mezclador mediante un módulo Ethernet de Arduino [3].

El transmisor dispone de un módulo de radiofrecuencia, que emite en las bandas de 915 y 433MHz. Este se comunica con unos dispositivos receptores que cuentan con tres ledes en posiciones diversas, que proporcionan información de *tally* y del estado del dispositivo. El receptor, que dispone de batería, puede alimentarse por micro-USB y tiene un switch para configurar el número de la cámara al que irá asociado. Finalmente, una interfaz web en HTML sirve para realizar la configuración de red necesaria para establecer conexión a nivel IP con el mezclador.

### 2.1.5. FLEXTALLY, CEREVO

El sistema de Cerevo es compatible con multitud de mezcladores de diferentes fabricantes. La estación dispone de dos entradas para dar cobertura a todos ellos, una de ellas es un GPIO y la otra un puerto Ethernet, utilizando esta última para su conexión con el ATEM.

La conexión con las lámparas puede hacerse mediante par trenzado y estándar RS-485 o bien por radiofrecuencia en las bandas de 433MHz (EUA/EU) y 315MHz (Japón) [4].

El suministro de energía en ambos dispositivos es mediante micro-USB, aunque la lámpara contiene una batería con una autonomía de 6 horas.

Además, dispone de una aplicación que corre en Windows, para configurar la conexión con el mezclador cuando se hace por medio de Ethernet.

Tanto la estación como las lámparas tienen micro interruptores eléctricos DIP de 8 posiciones para configurar el funcionamiento de estos. En la estación, determinan el modo de conexión con el mezclador, el ID de la estación y el tipo de conexión con la lámpara entre otras opciones. En la lámpara, configuran el canal de radiofrecuencia, el ID de la estación y el modo de conexión con esta.

Por último, los dispositivos disponen de un indicador led de encendido que en las lámparas proporcionan, además, información del estado de la batería. Así mismo, las lámparas tienen un botón para regular el brillo de la misma.

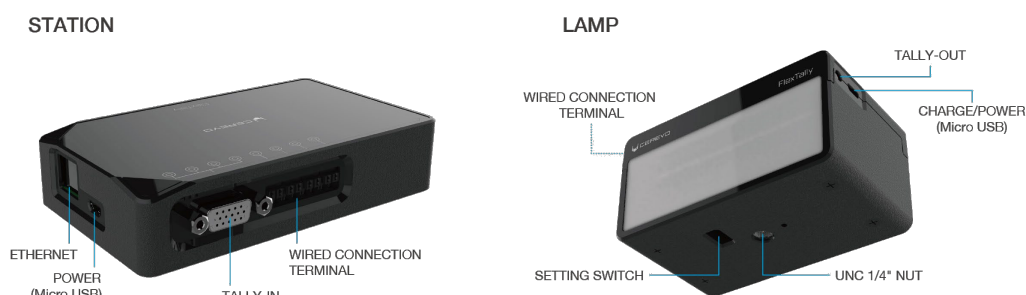


Ilustración 4. Estación y lámpara de CEREVO. Fuente: <https://flex tally.cerevo.com/>.

### 2.1.6. CUEBI wireless tally system

La solución de Cuebi, también ofrece compatibilidad con múltiples mezcladores. La particularidad de esta es que una aplicación disponible para MacOs y Windows proporciona la interfaz para establecer la conexión con el mezclador. Además, permite configurar parámetros como el canal de la red Wi-Fi a 2.4Ghz por el que, mediante UDP (*User Datagram Protocol*) multicast, se transmitirá la señal. Requiere que se conecte un punto de acceso al mezclador y de una alimentación de 5V por medio de micro-USB.



Ilustración 5. Receptor Cuebi. Fuente: <https://www.cuebi.com>.

Por otro lado, los dispositivos de Cuebi disponen de un método de accionamiento de indicadores externos. Este método se selecciona a través de la aplicación y necesita de un cable en Y, con un conector USB-A para la fuente de alimentación, un micro-USB para conectar a la Cuebi lighth y un mini-jack de 3.5mm TRS que en la punta portará la señal de 3.3V de programa y el anillo la de anticipo [5].

### 2.1.7. TALLY-LIGHTS.

La compañía Tally-lights, LLC dispone de diferentes controladores, que obtienen la señal de *tally* vía Ethernet y la distribuyen mediante cable de audio con adaptador jack de 3.5mm o de manera inalámbrica a sus propias luces de *tally*. No hay ninguna información del funcionamiento más allá del que se deduce por sus entradas y salidas.

## 2.2. Crítica al estado del arte

Como se ha expuesto en el punto anterior, existen en el mercado diferentes soluciones para llevar la señal desde un mezclador ATEM a los operadores de cámara. Sin embargo, el precio de la mayoría de estos sobrepasa los mil Euros, siendo este excesivo para pequeñas producciones, puesto que es un elemento que mejora pero no limita una grabación. Además, algunas de estas soluciones incluyen en su esquema sistemas de intercomunicación o cámaras, duplicando dicho material si ya se dispone de este.

Por otro lado, propuestas como las de Kvitko se hacen bajo pedido previo, puesto que no venden un producto, sino que ofrecen un servicio de ensamblaje de un kit de código abierto basado en el proporcionado por SKAARHOJ.

La propuesta de Cerevo no solo es la más económica, sino que ofrece compatibilidad con multitud de mezcladores, es independiente de las cámaras utilizadas, portable y puede usarse a través de conexiones inalámbricas o bien por cable. Sin embargo se trata de un producto cuyo hardware y software son privativos, por lo que el uso y modificación del mismo queda restringido.



### 3. Propuesta

Quiere realizarse un sistema funcional basado en código abierto, puesto que estamos convencidos de lo que esto conlleva. En general, cuando el código fuente y otros elementos son de dominio público, tan solo hay escasez de las habilidades requeridas para emplear de manera productiva dichos recursos por lo que pueden venderse servicios relacionados con el software libre, existiendo así una comunión entre el acceso universal y la rentabilidad comercial. En este sentido van a utilizarse las librerías C++ compatibles con el entorno de desarrollo Arduino, creadas por SKAARHOJ como punto de partida para desarrollar un sistema de *tally* para el mezclador ATEM, independiente del modelo de las cámaras y cuyo coste de fabricación sea ostensiblemente menor que el precio de mercado de los sistemas actuales.

Un primer dispositivo se conectará mediante Ethernet directamente al mezclador o bien por medio de un conmutador de red (switch) o un concentrador (hub), del que recibirá información en tiempo real sobre su estado. Esta estación base enviará el estado de la señal de programa y anticipo a las estaciones receptoras, que integrarán dos indicadores luminosos que responderán en función de dicha señal y de la cámara a la que están asociados. Se utilizará un interruptor eléctrico DIP para asociar una estación a una cámara y un potenciómetro para regular la intensidad de la luminosidad de los indicadores, adaptándola así a las condiciones del entorno.

Ambos dispositivos, utilizarán placas de desarrollo Arduino compatibles. El primero de ellos, será un WEMOS D1 R1 con un módulo ESP8266 integrado, al que se le acoplará un módulo Ethernet externo que integra un chip Wiznet W5100. Por otro lado, las estaciones receptoras se implementarán utilizando placas de desarrollo Lolin NodeMcu 1.0 V3. La comunicación entre ambos será inalámbrica y se realizará a través de tecnología Wi-Fi basada en el estándar IEEE 802.11 y operará en la banda de 2.4GHz. En una primera solución, la propia WEMOS actuará como punto de acceso para generar la red Wi-Fi. Como segunda opción, se implementará un punto de acceso independiente sobre una placa de desarrollo NodeMcu adicional.

El sistema constará de una estación base y tres receptoras, que darán soporte a las tres cámaras de las que dispone actualmente el grado en Ingeniería de Sistemas Audiovisuales impartido en la ESEIAAT. Estas cámaras tienen un conector USB, que proveerá de alimentación a los dispositivos receptores mediante un cable USB A/A, aunque también podrá conectarse a la alimentación de red eléctrica utilizando una fuente AC/DC de 5V y 500mA, o una batería externa, ambos con adaptador micro-USB. Un *loop* USB volverá a proveer a la cámara de dicho conector para poder conectar otros dispositivos.

En cuanto a la estación base, esta se alimentará por medio de un adaptador micro-USB, a través de un dispositivo de la mesa de realización que disponga de conector USB libre, de una fuente de alimentación de red con un adaptador AC/DC de características similares a la utilizada para alimentar las estaciones receptoras, o bien con una batería externa.

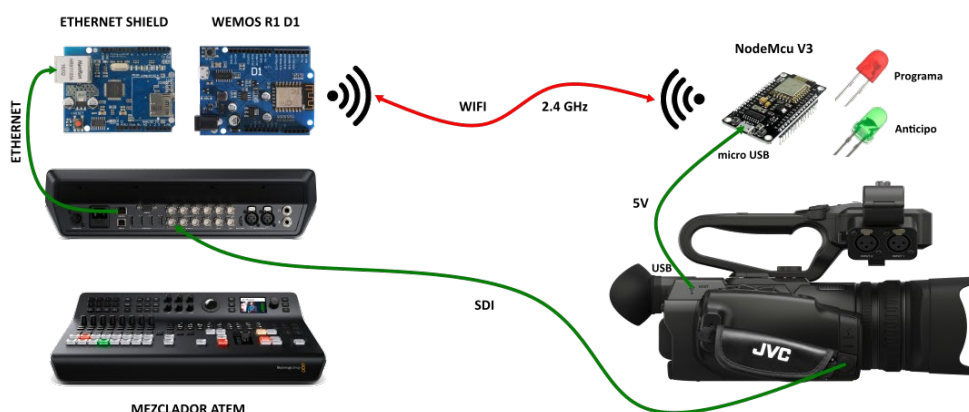


Ilustración 6. Esquema técnico propuesto (sin punto de acceso independiente). Fuente propia, Abril 2019.

Para la solución alternativa, se añadirá un punto de acceso externo que genere la red Wi-Fi a la que se conectarán las estaciones base y receptoras. Esto, junto a la posibilidad de ubicar la estación base cerca de las receptoras con una tirada de cable Ethernet, limitará las situaciones en las que el sistema pueda tener problemas de conectividad.

Se utilizará Arduino IDE como software para programar los microcontroladores de las placas señaladas.

### 3.1. Decisión sobre soluciones alternativas

#### 3.1.1. Medio de transporte de la señal de tally

Un punto crítico, que tendrá fuertes implicaciones en las características finales del sistema, es el de cómo llevar la información desde el dispositivo que obtiene la información de tally hacia las estaciones. La primera disyuntiva es la elección de un sistema inalámbrico o cableado. En ambos sistemas, hay ciertos factores que afectan negativamente al paso de una señal de un punto a otro y todos ellos lo hacen de manera más acusada en los sistemas inalámbricos.

- Uno de estos factores son las interferencias electromagnéticas causadas por campos electromagnéticos (EMI) generados por fuentes de radiación que operan en el mismo rango o próximos entre sí. En el caso de la tecnología Wi-Fi, las interferencias más problemáticas estarán causadas por la existencia de otros puntos de acceso situados en el mismo canal o en

los canales adyacentes. Dado que tan solo pueden usarse 12 de los canales del estándar IEEE 802.11, es común encontrarse en la situación antes descrita.

- Otro factor es la atenuación presente en cualquier medio de transmisión, que hace que la energía de la señal decaiga con la distancia. En medios guiados, esta reducción de la energía es por lo general exponencial y, por lo tanto, se expresa generalmente como un número constante en decibelios por unidad de longitud. En medios no guiados, la atenuación es una función más compleja de la distancia y es dependiente, a su vez, de las condiciones atmosféricas.

Para un cable Ethernet tipo par trenzado, la distancia máxima entre dos nodos sin necesidad de estaciones repetidoras es de unos 100 metros, aumentando a los 185m utilizando un cable coaxial (10Base2) y alcanzando hasta los 5000m con la fibra óptica (1000BaseLX).

En cuanto a la tecnología Wi-Fi, a una frecuencia de 2.4 GHz la pérdida de potencia a una distancia de 100 metros es de aproximadamente 80dB, asumiendo que no hay efecto Fresnel y sin presencia de obstáculos [6].

$$L_{bf}[dB] = 20 \log \left( \frac{4\pi \cdot d \cdot f}{c} \right), \text{ donde } d \text{ es la distancia en Km y } f \text{ la frecuencia en GHz.}$$

El chip ESP8266 radia con una potencia ( $P_r$ ) de 17dBm y una sensibilidad de recepción ( $P_L$ ) de -75dBm para el estándar IEEE 802.11 g (con una velocidad de conexión de 11Mbps), así como una ganancia de la antena de 2dBi ( $G_T$  y  $G_R$ ) [7]. Esto conlleva que, a una distancia de 100m, el margen teórico es de unos 16dB, por encima de los 10dB que debe tener un enlace [8].

17 dBm ( $P_r$ , Potencia radiada)  
 + 2 dBi ( $G_T$ , Ganancia antena Tx)  
 + 2 dBi ( $G_R$ , Ganancia antena Rx)

21 dBm  
 -80 dB ( $L_{bf}$  a 0.1Km)

-59 dBm (Nivel de señal recibido esperado)  
 --75 dBm ( $P_L$ , Sensibilidad Rx)

16 dB (Margen del enlace)

$L_0$  (Pérdidas básicas de la Ecuación de Friis) =  $P_r + G_T + G_R - P_L = 96$  dB

$$\text{rango (Km)} = 10^{\frac{L_0 - 32.5 - 20 \log(f(\text{MHz}))}{20}} \approx 0,62 \text{ Km}$$

Así, según las especificaciones del ESP8266, el rango teórico máximo del radioenlace en el espacio libre es de unos 600 metros. Estos cálculos no tienen en cuenta los objetos o personas que en una situación real sí estarían presentes, por lo que el margen del enlace y el rango disminuirían considerablemente.

Otro punto a considerar es el tiempo de montaje. Un sistema inalámbrico configurado previamente tan solo ha de acoplarse a la cámara, evitándose una tirada de cable por cada una de las estaciones.

También deberá considerarse la seguridad de un sistema por cable en verso a un sistema inalámbrico sujeto a ataques. Por último, la velocidad de transmisión, pese a ser considerablemente inferior en sistemas inalámbricos, no será un elemento crítico dadas las características de nuestro flujo de datos.

Una vez analizadas las implicaciones generales de ambas opciones, estas deben contextualizarse para determinar si son críticas para el correcto desarrollo de nuestra actividad. Los mezcladores ATEM de los que dispone la Universidad, tienen un uso básicamente formativo y se utilizan casi de manera exclusiva en un plató de menos de 100m<sup>2</sup>. En ocasiones, se realizan sesiones formativas en diferentes espacios como el Conservatori de Terrassa, el Teatre Principal o el Recinte Firal de Terrassa. Teniendo esto en cuenta, se ha decidido que es primordial reducir al máximo las tiradas de cableado entre equipos, disminuyendo así el tiempo invertido en el montaje y el coste final del producto.

Dado que existen soluciones para solventar los posibles problemas de interferencias y atenuación descritos, se ha optado por utilizar la tecnología Wi-Fi como medio de interconexión entre nuestros dispositivos. Se han descartado otras opciones inalámbricas como la radiofrecuencia a 315/433MHz, ya que la comunicación es unidireccional, implican añadir módulos adicionales a las placas de desarrollo y aumenta por tanto el coste final.

Se ha descartado la opción de embeber la señal de *tally* con la de audio de los intercomunicadores, que no implicaría cableado adicional, por el coste y la complejidad adicional que supone modular y demodular la señal, quedando como tarea para trabajos futuros.

### 3.1.2. Placas de desarrollo

La elección de la tecnología Wi-Fi, encamina a la utilización de placas de desarrollo con un periférico que disponga de esta. Se descartan las placas que incluyen periféricos como el WINC1500, que necesitan que sea el microcontrolador quien corra el código de bajo nivel asociado a las funciones de red, consumiendo memoria RAM de este.

Al inicio del proyecto, se dispone de dos placas de desarrollo, la WEMOS D1 R1 y la Lolin NodeMcu V3. Ambas integran un módulo ESP12E con el SoC (system on chip) ESP8266 de la compañía Espressif. Este chip integra un procesador Tensilica L106 de 32-bits [7], con conectividad Wi-Fi mediante estándar 802.11 b/g/n a 2.4GHz y que permite al usuario cargar su software en el mismo CPU en el que corre la

pila Wi-Fi/TCP/IP. La placa WEMOS está diseñada para simular el patillaje de la placa Arduino Uno, pudiendo acoplarse al Ethernet Shield del que se dispone.

Para iniciar cuanto antes la realización de los *sketches* de obtención, transmisión y recepción de la señal de *tally*, se decide utilizar las placas y módulos de los que se dispone. Dados los buenos resultados de ambos, no se valora la utilización de otras placas como la Adafruit HUZAH, que a su vez tienen un coste superior pese a montar el mismo SoC.

## 4. Desarrollo software

### 4.1. Estudio de las librerías de SKAARHOJ

Escritas en lenguaje C++ y compatibles con el software IDE de Arduino, nos permitirán establecer conexión con el mezclador a través de Ethernet y disponer así de la señal de *tally* en tiempo real. Según sus autores, son compatibles con versiones del ATEM Control Panel hasta la 7.5.0 [2].

De entre todas las librerías que pone a disposición SKAARHOJ, entre las que se encuentran las que utilizan todos sus productos, nos centraremos en las que desgranar el funcionamiento de los mezcladores ATEM. La SKAARHOJ Arduino ATEM Library contiene todas las clases, objetos y funciones necesarias para establecer conexión, obtener y controlar los parámetros de los mezcladores de video ATEM. Con este propósito, existen hasta 7 versiones de dicha librería, que difieren básicamente en su extensión y, por ende, en la cantidad de parámetros que abarcan. La librería ATEMstd contiene las funcionalidades básicas, optándose por empezar a revisar su contenido como primera opción. Así, obtenemos los constructores y funciones necesarias, que se analizan a continuación:

#### 4.1.1. Constructor objeto de la clase ATEM.

Puede inicializarse sin argumentos, utilizando la función *begin()* para ello.

```
ATEM (const IPAddress ip, const uint16_t localPort);
ATEM::ATEM (const IPAddress ip, const uint16_t localPort) {
    _ATEM_FtbS_state = false;           // Estado del fundido a negro (Fade To Black State)
    begin(ip, localPort);              // Llamada a la función begin()
}
```

#### 4.1.1.1. Funciones o atributos públicos de la clase ATEM.

La siguiente función inicializa dos atributos del objeto ATEM, la dirección IP y el puerto local del mezclador desde el que enviar paquetes.

```
void ATEM::begin(const IPAddress ip, const uint16_t localPort){
    EthernetUDP Udp;                       // Inicializamos objetos de comunicación
    _Udp = Udp;
    _switcherIP = ip;                      // Inicializa el atributo correspondiente a la dirección IP del mezclador
    _localPort = localPort;                // Inicializa el atributo puerto local
    _serialOutput = false;
    _isConnectingTime = 0;
    _ATEM_AMLv_channel=0;
    Serial.println("Preparating Connection...");
}
```

La función *connect()* inicia la conexión (*handshake*) con el mezclador. Para ello se inicializa un puerto local desde el que enviar datagramas UDP a la IP y puerto del mezclador especificados en la función

anterior. Puede observarse que se utiliza el puerto 9910 del mezclador como puerto remoto en la función *beginPacket()* de la librería *EhernetUDP*. A este puerto se enviará el *array* de bytes *connectHello[]*, vital para establecer y mantener la conexión con el mezclador.

```
void ATEM::connect() {
    Serial.println("Connecting...");
    _isConnectingTime = millis(); // milisegundos desde que se inició el programa actual
    _localPacketIdCounter = 1; // Inicializa variable localPacketIDCounter a 1
    _hasInitialized = false;
    _lastContact = 0;
    _Udp.begin(_localPort); //Puerto por el que escucharemos, retornando 1 si hay sockets disponibles
    _lastContact = millis();
    if (_serialOutput) {
        Serial.println(F("Sending connect packet to ATEM switcher."));
    }
    byte connectHello[] = { 0x10, 0x14, 0x53, 0xAB, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3A, 0x00, 0x00,
    0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
    _Udp.beginPacket(_switcherIP, 9910); //Inicia una conexión para escribir datos UDP a una conexión remota
    Serial.println("Beginning Packet");
    _Udp.write(connectHello,20); // Se envían los 20 bytes del mensaje connectHello
    Serial.println("Packet already wrote");
    _Udp.endPacket(); // Retorna 1 si el paquete se envió correctamente, 0 si hubo error
    Serial.println("Packet already sended");
}
}
```

La función *runLoop()* envía paquetes en respuesta a los mensajes de interrogación de presencia que envía periódicamente el mezclador. Esta función debe invocarse con una cadencia mínima de 2 veces por segundos dentro de la función principal *loop()*, de lo contrario el mezclador finaliza la conexión, dejándose de recibir la información de tally a través de la red.

```
void ATEM::runLoop() {
    uint16_t packetSize = 0;
    if (_isConnectingTime > 0) {
        packetSize = _Udp.parsePacket(); // Chequea la presencia de paquetes UDP y devuelve su tamaño
        if (_Udp.available() && packetSize==20) { // El tamaño de la respuesta del ATEM es de 20 bytes
            _Udp.read(_packetBuffer,20); // Leemos el paquete de 20 bytes
            _sessionId = _packetBuffer[15]; // Extraemos el ID de la sesión _Udp.beginPacket(_switcherIP, 9910);
            byte connectHelloAnswerString[] = {0x80, 0x0c, 0x53, 0xab, 0x00, 0x00, 0x00, 0x00, 0x00,
            0x03, 0x00, 0x00};
            _Udp.write(connectHelloAnswerString,12); // Se escribe una respuesta al mezclador
            _Udp.endPacket(); // Se envía dicha respuesta mediante UDP
            _isConnectingTime = 0; // Fin de la conexión, se reinicia la variable isConnectinTime a 0
        } else {
            if (_isConnectingTime+2000 < (unsigned long)millis()) {
                if (_serialOutput) {
                    Serial.println(F("Timeout waiting for ATEM switcher response"));
                }
                _isConnectingTime = 0;
            }
        }
    } else { // Si hay datos disponibles, se leen y se vacía el buffer
        while(true) {
            packetSize = _Udp.parsePacket();
            if (_Udp.available() && packetSize !=0) {
                _Udp.read(_packetBuffer, 12); // Se leen los 12 bytes correspondientes al header del paquete
                uint16_t packetLength = word(_packetBuffer[0] & B00000111, _packetBuffer[1]);
                _lastRemotePacketID = word(_packetBuffer[10], _packetBuffer[11]);
                uint8_t command = _packetBuffer[0] & B11111000;
                boolean command_ACK = command & B00001000 ? true : false;
                boolean command_INIT = command & B00010000 ? true : false;
            }
        }
    }
}
```





de coincidencia, se extrae cierta posición de *packetBuffer[]*, que variará en función de si la versión del *firmware* del mezclador es inferior a la versión 2.0.

```
void ATEM::_parsePacket(uint16_t packetLength) {
...
  char cmdStr[] = { _packetBuffer[4], _packetBuffer[5], _packetBuffer[6], _packetBuffer[7], '\0'};

  if(strcmp(cmdStr, "PrgI") == 0) { //strcmp(string1, string2) compara entre dos strings
    if (!ver42()) { // Versiones inferiores a la 2.0 del firmware
      _ATEM_PrgI = _packetBuffer[1];
    } else { // Versiones igual o superiores a la 2.0 del firmware
      _ATEM_PrgI = (uint16_t)(_packetBuffer[2]<<8) | _packetBuffer[3];
    }
  }
  ...
}

bool ATEM::ver42() // Función que retorna true si la versión del firmware es la 2, la 2.12 o superior
  return (_ATEM_ver_m>2) || (_ATEM_ver_m>=2 && _ATEM_ver_l>=12);
}
```

Como puede observarse, se incluye la librería *EthernetUdp*, que será necesaria para el envío de paquetes mediante UDP. Además, en el documento *ATEMstd.cpp* se proporciona información de interés que puede ser crítica para el buen funcionamiento del script a elaborar. Por ejemplo, en este se menciona que el ATEM envía información entre 10 y 20kbytes de datos sobre el estado del sistema, en particular cuando este se inicia. En cambio, el buffer de recepción de la interfaz Ethernet del *Arduino Shield* es de 2kbytes, por lo que tan solo se recibe correctamente el primer paquete. La información importante está en dicho paquete, por lo que la pérdida de esta no es crítica.

#### 4.1.2. Función *setup\_ethernet* de la clase *ATEMTally*.

Esta función es necesaria para establecer la conexión con el mezclador. Precisa de la inicialización de un objeto *ATEMTally* y posteriormente se realiza la llamada a la función , dentro de la función *setup()*.

```
void ATEMTally::setup_ethernet(byte mac[6], byte ip[4], byte switcher_ip[4], int& switcher_port) {
  int idcheck = EEPROM.read(0);

  if (idcheck == ID) { // Si la memoria EEPROM contiene datos guardados, se obtiene la mac, la IP y la IP del mezclador

    for (int i = 0; i < 6; i++){
      mac[i] = EEPROM.read(i+1);
    }
    for (int i = 0; i < 4; i++){
      ip[i] = EEPROM.read(i+7);
    }
    for (int i = 0; i < 4; i++){
      switcher_ip[i] = EEPROM.read(i+11);
    }
    // Lee el puerto del mezclador (2 bytes) desde las direcciones 15 y 16
    switcher_port = ATEMTally::eeprom_read_int(15);
  }

  Ethernet.begin(mac, ip); // Se inicia la conexión Ethernet con los datos obtenidos
}
```

## 4.2. Desarrollo software de la estación base

Este software permite al microcontrolador las siguientes funcionalidades:

- Establecer y mantener la conexión con el mezclador.
- Obtener la información de *tally* del mezclador.
- Levantar un punto de acceso y una estación Wi-Fi para enviar esta información a los microcontroladores receptores, ubicados junto a las cámaras.
- Elegir el canal del protocolo IEEE 802.11 b/g/n con menos interferencias producidas por otros puntos de acceso.

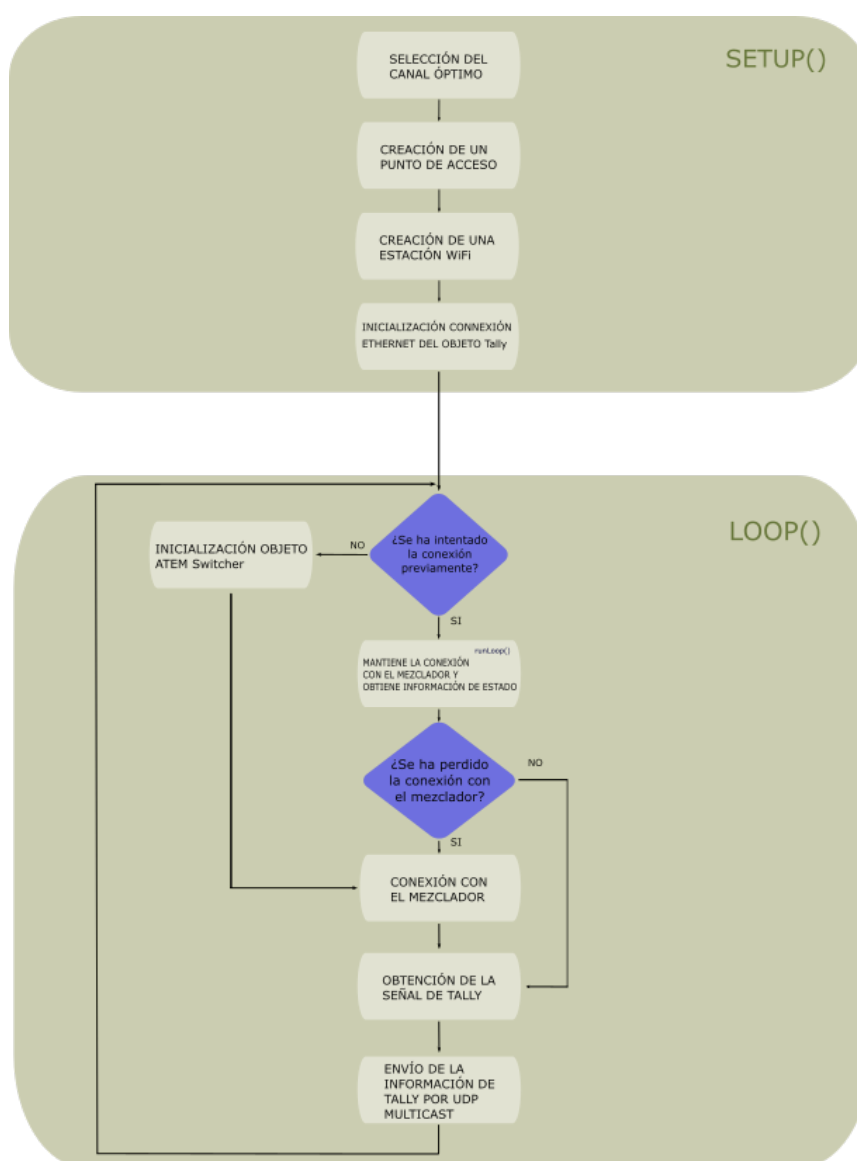


Ilustración 7. Diagrama de bloques de la estación base con punto de acceso propio. Fuente propia, Junio 2019.

#### 4.2.1. Implementación de las librerías SKAARHOJ con Arduino IDE.

Una vez estudiadas las librerías pertinentes, se realiza un *sketch* inicial utilizando las funciones descritas, con el único objetivo de obtener la información de *tally* a través de Ethernet. Para ello contamos con el software de Arduino (IDE), en su versión 1.8.8. Se dispone de una placa Arduino duemilanove, cuyo microcontrolador está basado en el Atmega328 [9] y de un WEMOS D1 R1 con un módulo ESP8266. Ambos son compatibles con un módulo Ethernet que integra un chip Wiznet W5100 que provee de una pila de red IP capaz de soportar TCP y UDP. El sketch resultante es el siguiente:

```
#include <ATEM.h>
#include <EthernetUdp.h>

byte switcher_ip[] = {192,168,10,241}; // IP del mezclador
int switcher_port = 49910; // Puerto del mezclador
ATEM AtemSwitcher; // Se genera un objeto AtemSwitcher
ATEMTally AtemTally; // Se genera un objeto AtemTally
boolean doneOnce = false;
struct // Se define una estructura para la información de tally
{
    int program, preview;
} payload;

void setup() {
    Serial.begin(921600); // Inicializamos el puerto serie
    ATEMTally.setup_ethernet(mac, ip, switcher_ip, switcher_port); // Inicia la conexión Ethernet con el ATEM
}

void loop(){
    if (!doneOnce) { // Tan solo se ejecuta una vez
        AtemSwitcher.begin(IPAddress(switcher_ip[0], switcher_ip[1], switcher_ip[2], switcher_ip[3]),
            switcher_port); // inicializamos el objeto AtemSwitcher
        Serial.println("AtemSwitcher.begin"); // Impresión de control
        AtemSwitcher.connect(); // Intento de conexión con el mezclador
        Serial.println("AtemSwitcher.connect"); // Impresión de control
        doneOnce = true;
    }
    AtemSwitcher.runLoop(); // Obtiene información del estado del mezclador y mantiene la conexión con este
    Serial.println(" AtemSwitcher.runLoop"); // Impresión de control
    if (AtemSwitcher.isConnectionTimedOut()) { // Se procede a reconectar si se ha perdido la conexión
        AtemSwitcher.connect();
        Serial.println("AtemSwitcher.connect (runLoop)"); // Impresión de control
    } else { // Se asigna la información de tally en la estructura creada
        payload.program = AtemSwitcher.getProgramInput();
        payload.preview = AtemSwitcher.getPreviewInput();
        Serial.println(payload.program); // Impresión de control
        Serial.println(payload.preview); // Impresión de control
    }
    delay(300); // Introducimos un pequeño retraso para poder leer las impresiones
}
```

Deberán añadirse las librerías utilizadas para poder subir el sketch a la placa de manera satisfactoria. Para ello, se accede a la opción "incluir librería" de la pestaña "Programa" y a continuación, incluimos la biblioteca .ZIP. Seguidamente, se configura el compilador de Arduino accediendo a la pestaña "Herramientas", seleccionamos la placa Arduino Duemilanove y el procesador ATmega328P. Tras conectar la placa mediante USB, esta es detectada por el sistema operativo (Windows 10 Home) y se

instalan los drivers necesarios de manera automática. Por último, se asigna el puerto que aparece por defecto, se abre un monitor del puerto serie y se realiza la subida del sketch de manera satisfactoria.

Para verificar el funcionamiento del sketch, se acopla el módulo de Ethernet al Arduino Duemilanove y se conecta al mezclador mediante un cable Ethernet. En cuanto se enciende el mezclador, observamos las líneas de control en el monitor Serie, incluidas las pertenecientes a la información de *tally*. El tiempo de respuesta entre que se presionan las teclas del mezclador y se imprime en el monitor Serie el número de las cámaras es inapreciable, por lo que a priori, no será un factor que influya negativamente en la dinámica de realización.

A continuación, se realiza una segunda prueba, esta vez con la placa WEMOS D1. Para ello, primero debe instalarse el *plug-in ESP8266* en caso de que el IDE Arduino instalado no disponga de este. Accediendo a Archivo/Preferencias, se añade la siguiente línea en el gestor de URL's adicionales de tarjetas:

[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

Esto propiciará que el *plug-in* esté disponible para instalar en el gestor de tarjetas que hay en Herramientas/Placa. Una vez instalado podrá seleccionarse placa WEMOS D1 R1 que, a diferencia de la Arduino, tiene la posibilidad de modificar ciertos parámetros. Se seleccionará la frecuencia mínima de la CPU (80 MHz) y la velocidad de subida a 921600 tal y como se indica en la propia placa. Si se desea imprimir datos a través del puerto serie, deberá iniciarse este con la velocidad de transmisión que coincida con la velocidad de lectura del puerto serie. Por último, se ha cargado la librería Ethernet específica para el chip ESP8266, ya que la genérica no es compatible.

## 4.2.2. Implementación de la conectividad Wi-Fi

Mediante la utilización del modo WIFI\_AP\_STA (`WIFI.mode(WIFI_AP_STA)`), el ESP8266 puede conmutar entre el modo WiFi\_AP y el WiFi\_STA [10]. De esta manera, en un único microcontrolador se levantará un punto de acceso y, a posteriori, una estación Wi-Fi que se conectará a este. La estación Wi-Fi será la encargada de enviar la información de *tally* recibida del mezclador, hacia todas las estaciones receptoras. Para ello, se utilizará el protocolo de nivel de transporte orientado a mensajes UDP multicast. Este modelo de funcionamiento conlleva que el envío de mensajes vía UDP no sea constante, debido a que ambos modos (AP y STA) no coexisten de manera simultánea, afectando al tiempo de respuesta de los LEDs. Este comportamiento ha sido detectado capturando el tráfico UDP a través de la red Wi-Fi con el software Wireshark 2.6.4. Como puede observarse en la Ilustración 7, cuando el microcontrolador cambia al modo punto de acceso, este tiene varios paquetes para enviar, por lo que desde el primero de estos respecto al último enviado habrá transcurrido cierto intervalo tiempo. Este desfase marca el tiempo de respuesta de los LEDs y puede llegar a 1.7 segundos, aunque en general no alcanza 1 segundo, por lo que no resultaría crítico en la mayoría de casos.

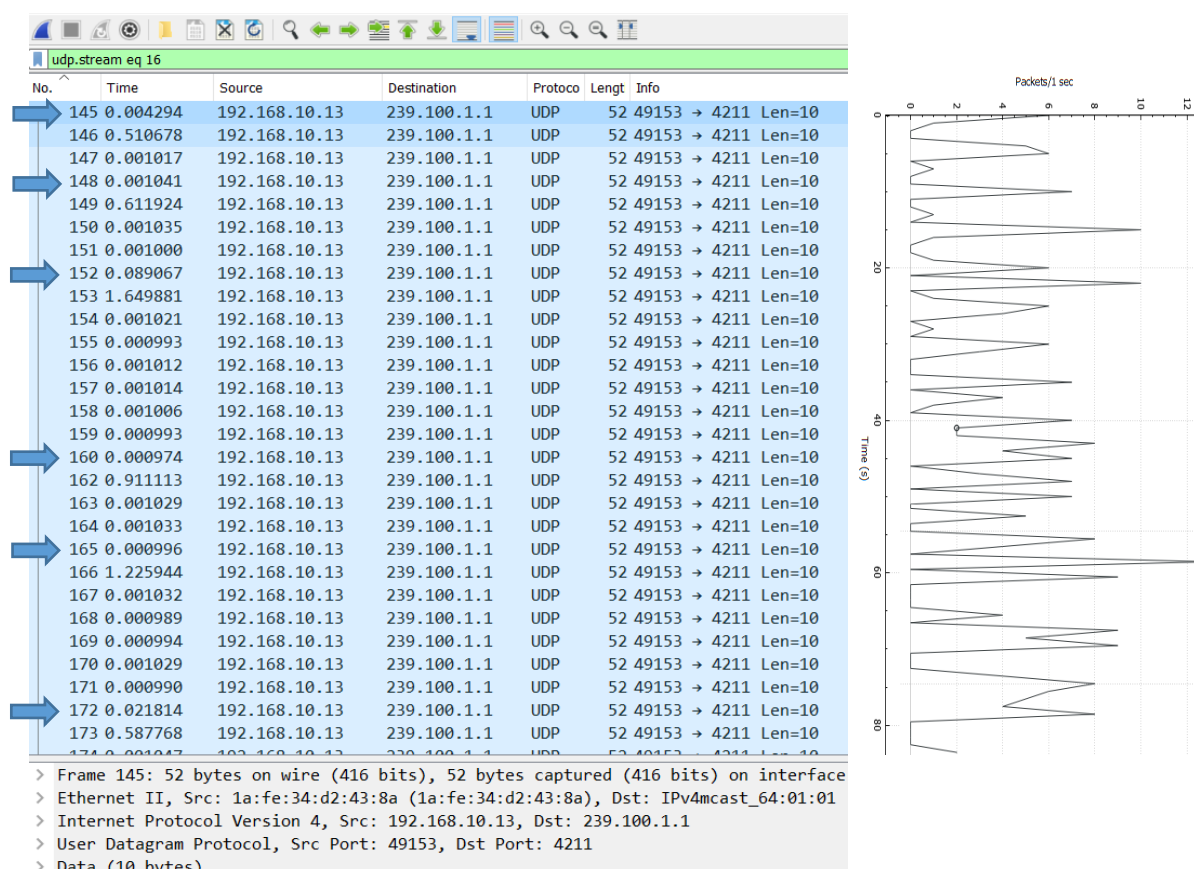


Ilustración 8. Captura del tráfico a través de la red WiFi formada con el punto de acceso creado por el ESP8266, con filtro por protocolo UDP, fuente y destino (izquierda). Gráfico de la tasa de paquetes UDP capturados según los filtros indicados (derecha). Fuente propia, Mayo 2019.

En caso de precisar de un tiempo de respuesta constante e inferior al que ofrece el modo WIFI\_AP\_STA, deberá realizarse la conexión a través de un punto de acceso externo. Para ello, se utilizará una placa de desarrollo NodeMcu adicional, que dispondrá únicamente del software para establecer un punto de acceso Wi-Fi y de selección de canal. Así, la estación base se conectará a este punto de acceso externo, para enviar la información de *tally* a las estaciones receptoras. Esta variante técnica implica la posibilidad de aumentar la distancia entre la estación base y las receptoras, puesto que el punto de acceso podría situarse entre ambos.

Por defecto, el ESP8266 utiliza la versión g de la norma 802.11, que tiene un rango, un ratio de transferencia y un consumo de corriente medios. Si se requiere, puede configurarse para tener un mayor rango a costa de disminuir el ratio de transferencia y aumentar también el consumo de corriente. Para ello debe utilizarse la siguiente sintaxis en el *setup()*, tras iniciar el punto de acceso [11]:

```
wifi.setphymode(wifi.PHYMODE_B);
```

#### 4.2.2.1. Establecimiento de un punto de acceso Wi-Fi

```
#include <ESP8266WIFI.h>
const char *APssid = "ESPsoftAP_01"; // Identificador de la red
const char *APpassword = "passT0p1!"; // Contraseña
IPAddress APlocal_IP(192, 168, 10, 13); // Dirección IP
IPAddress APgateway(192, 168, 10, 12); // Puerta de enlace
IPAddress APsubnet(255, 255, 255, 0); // Máscara
void setup() {
    WIFI.mode(WIFI_AP); // Modo punto de acceso
    WIFI.softAPConfig(APlocal_IP, APgateway, APsubnet) // Configura la red ( ip, puerta de enlace, máscara)
    WIFI.softAP(APssid, APpassword, 1) // Configura el punto de acceso (nombre, contraseña, canal)
    WIFI.softAPIP(); // Devuelve la dirección IP del interfaz de red del punto de acceso
}
```

#### 4.2.2.2. Establecimiento de una estación Wi-Fi

Añadimos al código anterior, las siguientes líneas.

```
IPAddress STAlocal_IP(192, 168, 10, 14); // Dirección IP de la estación
void setup() {
    WIFI.mode(WIFI_STA); // Modo estación
    Serial.printf("Station connecting to %s\n", APssid); // Impresión de control
    WIFI.begin(APssid, APpassword); // Inicializa la conexión con un punto de acceso (nombre , contraseña)
    WIFI.config(STAlocal_IP); // Configura una IP estática
    while (WIFI.status() == WL_DISCONNECTED){ //WIFI.status() retorna el estado de la conexión
        delay(5000);
        Serial.print(".");
    } // No saldrá del bucle hasta que se haya producido la conexión con el punto de acceso
    Serial.print("Station connected, IP address: "); // Impresión de control
    Serial.println(WIFI.localIP()); // Se imprime la dirección IP de la estación
    Serial.printf("Signal Strength: %d dBm\n", WIFI.RSSI()); // Se imprime la potencia de señal recibida (dBm)
}
```

### 4.2.3. Envío de la información de *tally* por UDP multicast

UDP es un servicio *best effort*, sin garantía de entrega o recepción, sin control de errores más allá de un *checksum* opcional. Tampoco dispone de control de flujo ni de congestión. Sin embargo a diferencia de TCP (*Transmission Control Protocol*), no introduce retraso por establecimiento de conexión, permite aplicaciones *multicast* e implica la utilización de recursos de procesamiento y memoria mínimos. El enrutado en Multicast se basa en la procedencia del paquete. La dirección IP destino no indica dónde debe reenviarse este, sino que está asociada con un grupo de receptores interesados. Esta tiene 32 bits y de acuerdo al RFC 2365 (del inglés: *Request For Comments*) el bloque de direcciones 239.0.0.0/8 está destinado a uso administrativo [12]. En concreto, nuestro sistema utilizará la dirección multicast 239.100.1.1 .

Para minimizar el impacto de las posibles pérdidas de paquetes, estos se enviarán repetidamente, a una tasa de 4 paquetes por segundo. Además, la información de *tally* no solo se enviará a los dispositivos asignados a las cámaras que sufren algún cambio en su estado. Esto hará que si una cámara pierde información sobre su estado y por tanto indica una información incorrecta, sea por el mínimo tiempo posible. UDP utiliza puertos que identifican a un proceso en un usuario, cuya longitud es de 16 bits. Para la utilización de este protocolo, se incluye la librería WiFiUdp.

```
#include <ESP8266WIFI.h> // Fichero de cabecera requerid por defecto si se utiliza el Wi-Fi del ESP8266
#include <WiFiUdp.h> // Fichero de cabecera usado para programar las rutinas UDP
IPAddress MCastIP(239, 100, 1, 1); // Dirección IP multicast
unsigned int localUdpPort = 4210; // Puerto local
unsigned int remotePort = 4211; // Puerto remoto
void setup() {
    Udp.beginMulticast(STALocal_IP, MCastIP, localUdpPort); // Inicia un socket UDP, en el puerto indicado
}
void loop() {
    byte tally []= {0,0,0,0,0,0,0,0}; // Inicializamos la variable tally
    payload.program = AtemSwitcher.getProgramInput();
    payload.preview = AtemSwitcher.getPreviewInput();
    if( payload.program == payload.preview){ // Modificamos la variable tally
        tally[payload.program] = 0x1;
    }else{
        tally[payload.program] = 0x1;
        tally[payload.preview] = 0x2;
    }
    Udp.beginPacketMulticast(MCastIP, 4211, WIFI.localIP()); // Inicia una conexión para escribir datagramas UDP
    Udp.write(tally, sizeof(tally)); // Escribe el datagrama UDP en la conexión remota
    Udp.endPacket(); // Finaliza el paquete y lo envía
}
```

Como puede observarse, se utiliza un array de bytes de 9 posiciones, donde la primera está reservada y las siguientes corresponderán a las 8 posibles entradas de las que dispone el mezclador. Dado que las funciones que obtienen la señal de anticipo y programa retornan el número de cámara correspondiente, esta se utiliza para acceder y modificar la posición del array.

#### 4.2.4. Selección del canal Wi-Fi

El estándar IEEE 802.11 dispone de 14 canales para comunicaciones Wi-Fi de baja potencia en la banda de radiofrecuencia ISM (del inglés: Industrial, Scientific and Medical) de 2.4GHz. La separación entre canales es de 5MHz (a excepción del canal 14, que está separado 12MHz respecto a su anterior), mientras que el ancho de banda típico requerido por canal es de 20 MHz, al que se le añade un intervalo de guarda de 2MHz [13]. Existe por tanto un solapamiento potencial en el rango de frecuencias operativo cuando dos transmisores operan en canales adyacentes en un mismo espacio aéreo.

Además, el uso de los canales 13 y 14 pueden estar prohibidos por las autoridades locales de cada país. Concretamente, en España no puede utilizarse el canal 14.

Hay estudios que indican que el receptor puede lidiar con otras señales WLAN (del inglés: *wireless local area network*) dentro de la misma banda [14]. El aislador de radiofrecuencia impide que señales de alto nivel de una fuente interfieran y creen productos de intermodulación en otra. Sin embargo, para minimizar el rechazo entre canales adyacentes y no adyacentes que estos propios estudios muestran, se implementa en el código del microcontrolador emisor un selector de canal. Con este, se pretende que el punto de acceso se establezca en el canal con menos interferencias intra e inter canal. Para ello, se utiliza la función *scanNetwork()*, que busca y retorna el número de redes Wi-Fi disponibles.

Después puede accederse a diferentes parámetros de cada una de estas, como la fuerza de la señal de un punto de acceso que nuestro dispositivo recibe, en dBm.

Para el cálculo de las interferencias, se ha tomado como referencia la forma de una señal IEEE 802.11 típica, como la que se muestra en la Ilustración 9.

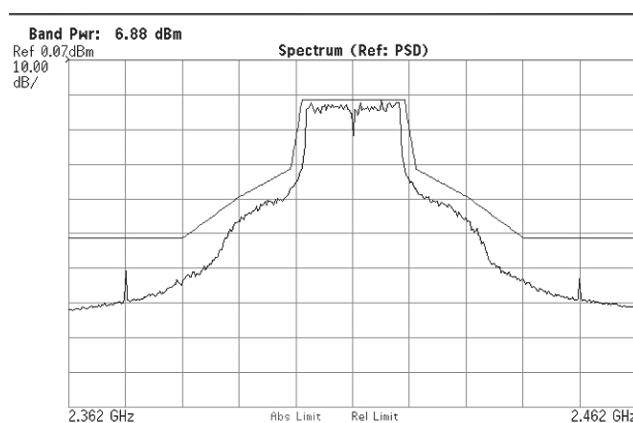


Ilustración 9. Señal IEEE 802.11 medida utilizando el analizador de espectro Keysight. Keysight Technologies IEEE 802.11 Wireless LANPHY Layer (RF) Operation and Measurement. Application note, 2017.

Así, si hay varias fuentes emitiendo en un mismo canal, se suma la potencia que se recibe de estas. A esta se le añade la potencia recibida de los canales inmediatamente adyacentes (no hay decaimiento



a 5 MHz de la frecuencia central). Por último se le suma una estimación de la potencia de las fuentes que emiten a 10 MHz de diferencia. Como puede observarse en el espectro de la señal, la potencia de esta decae con la frecuencia. En concreto, a 10 MHz de la frecuencia central, se observa que la potencia disminuye al menos 8 dB [15], por lo que para dicha estimación se tomará un valor superior, de 12dB.

$$fc(0) = a(-2)F(-2) + F(-1) + F(0) + F(1) + a(2)F(2)$$

Cada uno de los sumandos de la ecuación anterior corresponde a la interferencia sobre el canal evaluado de los canales contiguos.  $F(-2)$  es la potencia recibida a dos canales por debajo del actual y  $a(-2)$  es un factor de ponderación, que como puede observarse corresponde a 63,1mW (12 dB). A continuación, se muestra el código de la función que selecciona el canal donde establecer el punto de acceso:

```
int scanResult (int networksFound){

    float rssi_array[16]= {-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150};
    float rssi_array2[16]= {};
    float rssi, rssi_;
    float min_rssi = 999;
    int selection = -1;

    for (int i = 0; i < networksFound; i++){
        int pos = WiFi.channel(i)+1; // Posición de cada red encontrada en el array rssi
        rssi = WiFi.RSSI(i)/10.00; // dBm a mW paso 1
        float x = pow (10, rssi); // dBm a mW paso 2
        if (rssi_array[pos] == -150) {
            rssi_array[pos] = WiFi.RSSI(i);
        }else { // Suma fuentes incoherentes, sin correlación, con diferente offset de fase
            rssi_ = rssi_array[pos]/10.00;
            float y = pow(10, rssi_);
            rssi_array[pos] = 10 * (log10 (x+y)) ; // mW a dBm
        }
    }

    for (int j = 2; j < 14; j++){ // Se calculan las potencias recibidas de cada canal, añadiendo las interferencias
        // Caída de 12 dBm en la potencia de las señales dos canales a la izquierda
        float f1 = pow(10, (rssi_array[j-2]-12 / 10.00) );
        float f2 = pow(10, (rssi_array[j-1] / 10.00) );
        float f3 = pow(10, (rssi_array[j+1] / 10.00) );
        // Caída de 12 dBm en la potencia de la señal dos canales a derecha
        float f4 = pow(10, (rssi_array[j+2] -12 / 10.00) );
        float f0 = pow(10, (rssi_array[j] / 10.00) );
        rssi_array2[j] = 10 * ( log10 (f0 + f1 + f2 + f3 + f4) ); // Suma de fuentes incoherentes
    }

    for (int k = 2; k < 14; k++){ // Se busca el canal con la mínima potencia recibida
        if (rssi_array2[k] < min_rssi) {
            min_rssi = rssi_array2[k];
            selection = k-1;
        }
    }
    return selection;
}

void setup() {

    WiFi.scanNetworks();
    // Llamada a la función para obtener el canal seleccionado.
    int canal = scanResult(WiFi.scanComplete()); // Se utilizará en WIFI.softAP(APssid, APpassword, canal)
}
```

### 4.3. Desarrollo software de las estaciones receptoras

Permite a los dispositivos receptores obtener la información de *tally* enviada por la estación Wi-Fi del dispositivo emisor, encender y/o apagar dos ledes en función de la información anterior, modificar la intensidad luminosa de los ledes y gestionar el número de cámara al que va asociado el dispositivo.

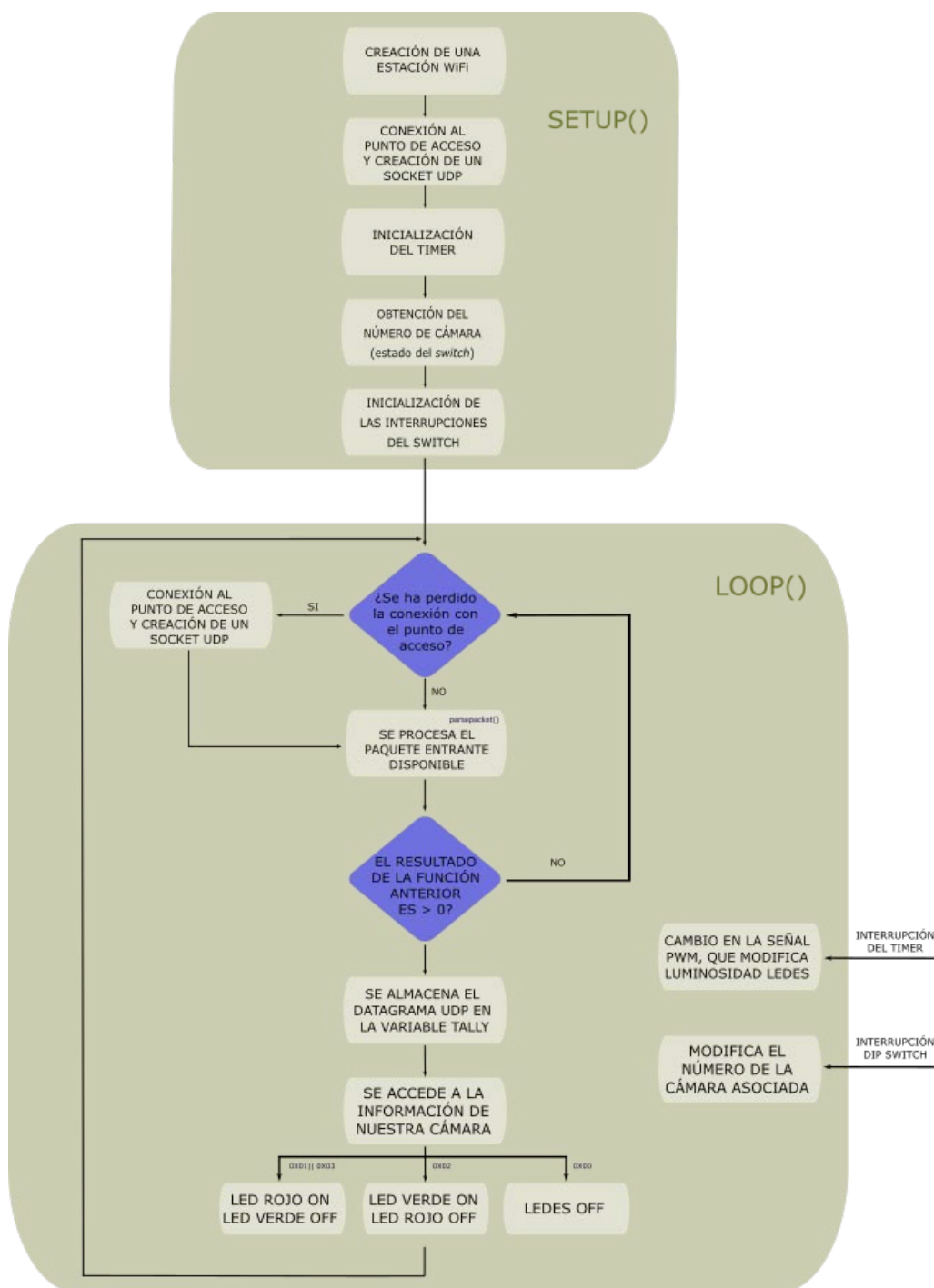


Ilustración 10. Diagrama de bloques de las estaciones receptoras. Fuente propia, Junio 2019.

### 4.3.1. Establecimiento de una estación Wi-Fi

Las estaciones receptoras actúan en modo WiFi\_STA, conectándose a un punto de acceso creado por el microcontrolador emisor o bien el que establece adicionalmente. Estas escanean los diferentes canales hasta encontrar el que distribuye la señal cuyo SSID (del inglés: *Service Set Identifier*) se ha marcado por software (en nuestro caso, ESPsoftAP\_01). En ocasiones el sistema entra en modo Modem-sleep para ahorrar energía. Espressif indica que el sistema se despierta de modo automático, gracias a que en todo momento mantiene la conexión al punto de acceso. Sin embargo, en ocasiones esto ocasiona ciertos problemas a la hora de establecer la conexión inicial. Para evitarlo se utiliza la siguiente interfaz, que inactiva dicho modo, después de incluir la librería *user\_interface*:

```
wifi_set_sleep_type(NONE_SLEEP_T);
```

### 4.3.2. Asociación a un número de cámara

La función *get\_cam\_adress()* lee el estado de los tres pines asociados a los micro interruptores del DIP. Esta lectura tiene 2<sup>3</sup> posibles resultados, coincidiendo con el máximo número de fuentes que puede gestionar el mezclador. El resultado de la función se usará para acceder a la posición del *array* que almacena la información de *tally* recibida.

Se establecen rutinas de interrupción asociadas a los pines del interruptor, que detectarán si se produce un cambio en el estado de los mismos para actualizar así el número de cámara.

```
int dipPins[] = {D3, D2, D1};
int cam_adress;

void get_cam_adress() {          // Modifica el número de cámara en función del estado de los pines del interruptor DIP
    int address;
    address = (address << 1) | (!digitalRead(D3));
    address = (address << 1) | (!digitalRead(D2));
    address = (address << 1) | (!digitalRead(D1));
    cam_adress = address;
}

void setup() {

    for(int i = 0; i<=2; i++){          // Se configuran los pines asociados al interruptor DIP
        pinMode(dipPins[i], INPUT_PULLUP);          // Se configuran como entradas
        digitalWrite(dipPins[i], HIGH);          // Se activa la resistencia interna
    }
    get_cam_adress();          // Obtenemos el número de cámara actual e inicializamos interrupciones para los pines del DIP
    attachInterrupt(digitalPinToInterrupt(D1), get_cam_adress, CHANGE);
    attachInterrupt(digitalPinToInterrupt(D2), get_cam_adress, CHANGE);
    attachInterrupt(digitalPinToInterrupt(D3), get_cam_adress, CHANGE);
}
```

### 4.3.3. Recepción de la información de tally por UDP multicast y gestión de los ledes

Una vez creado un objeto UDP e iniciada la conexión con el punto de acceso, se inicia un socket UDP en el puerto local, del mismo modo que en el dispositivo emisor.

```
Udp.beginMulticast(WiFi.localIP(),MCastIP,localUdpPort); // En el setup()
```

Después, en el bucle principal, se procesan los paquetes UDP entrantes. Si hay paquetes disponibles, se leerá el contenido de los mismos.

```
byte tally[];
void loop() {
  int packetSize = Udp.parsePacket(); //Devuelve el tamaño del paquete entrante, que será 0 si no hay disponibles
  if (packetSize>0) {
    Udp.readBytes(tally,UDP_TX_PACKET_MAX_SIZE); // Se almacenan los datos leídos en la variable tally
  }
}
```

Las estaciones receptoras obtendrán, a través de la red Wi-Fi, un paquete de datos con el estado de todas las cámaras. Cada dispositivo extraerá el estado de su cámara identificando la posición del byte de estado asociado a su cámara. Así el dispositivo de recepción número 1 (este identificador se selecciona a través del interruptor DIP), buscará el byte de estado asociado que corresponde al byte correspondiente a la segunda posición del array de bytes (el primer byte del paquete está reservado). El dispositivo de recepción número 2, identificará el estado a través del tercer byte y así sucesivamente.

Los valores posibles del byte de estado son:

- 0x00 : La cámara no está seleccionada (leds apagados)
- 0x01 : La cámara está seleccionada en programa (led rojo encendido)
- 0x02 : La cámara está seleccionada en anticipo (led verde encendido)
- 0x03 : La cámara está seleccionada tanto en programa como en anticipo (led rojo encendido)

#### 4.3.4. Regulación de la intensidad luminosa de un diodo electroluminiscente

La regulación se lleva a cabo mediante la utilización de señales con modulación por ancho de pulso PWM (del inglés: *Pulse Width Modulation*). Con ella, se pretende emular a una señal analógica, modificando la cantidad de energía que se envía al diodo. Es una señal cuadrada a la que se le cambia el *duty cycle*, que es el ancho relativo respecto el período de la misma. Así, simularemos un voltaje que será el valor promedio de la señal.

Para minimizar el consumo de recursos del microcontrolador, la generación de la señal PWM se ha realizado mediante interrupción. Esta provoca la deriva a una dirección específica de memoria, interrumpiéndose momentáneamente la ejecución del programa principal. A partir de esa dirección, se encuentra una subrutina que se encarga de realizar la operación de I/O (del inglés: *Input/Output*), devolviendo después al punto interrumpido del programa principal.

Concretamente se utiliza el timer1 de la librería Ticker [16], cuya resolución es de 16 bits (pudiendo contar hasta 65.536). El microcontrolador trabaja a 80MHz. Para disminuir esta frecuencia hasta los

5MHz, se utiliza un divisor del timer (TIM\_DIV16). Para lograr una frecuencia de 100Hz, que haga inapreciable el parpadeo de los ledes, el reloj deberá contar 50.000 veces.

El parámetro TIM\_SINGLE determina que debe establecerse un nuevo valor en la rutina de interrupción para iniciar el reloj de nuevo. Con ello, establece un tiempo de encendido del led y otro de apagado, que vendrá regulado por el estado del potenciómetro.

```
#include <Ticker.h>
#define T 50000
#define LED2 D0 // LED externo
boolean led_state = false;
int duty=0;
void ICACHE_RAM_ATTR onTimerISR() {
  led_state=!led_state;
  duty=analogRead(A0);
  if(led_state){
    timer1_write((duty*T)/1024);
  } else {
    timer1_write(((1024-duty)*T)/1024);
  }
  digitalWrite(LED2,!digitalRead(LED2));
}
void setup() {
  pinMode(LED2,OUTPUT);
  pinMode(A0,INPUT);
  digitalWrite(LED2,HIGH);
  timer1_isr_init(); // Inicializa el Timer1
  timer1_attachInterrupt(onTimerISR) // Se implementa la rutina de interrupción. Timer1 es un timer de 16 bits
  timer1_enable(TIM_DIV16, TIM_EDGE, TIM_SINGLE); // TIM_DIV16 => 5MHz (5 ticks/us - 1677721.4 us max)
  timer1_write(T); // Se llama a la rutina cada x ciclos de reloj (50000 = 10ms)
}
```

## 5. Desarrollo hardware

### 5.1. Desarrollo hardware de la estación base

El dispositivo emisor, consta de una placa de desarrollo WEMOS D1 R1 al que se le ha acoplado un módulo o *shield* Ethernet. Este último tiene un conector ICSP (del inglés: *In-Circuit Serial Programming*) de 6 pines [17], que expone la conectividad SPI (del inglés: *Serial Peripheral Interface*) entre la placa de desarrollo y sus periféricos, mientras que la placa WEMOS carece de este conector. Para solventarlo, deben cablearse los siguientes pines:

ICSP pin	WEMOS pin
1	12 (MISO)
3	13 (SCK)
4	11 (MOSI)

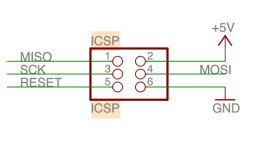
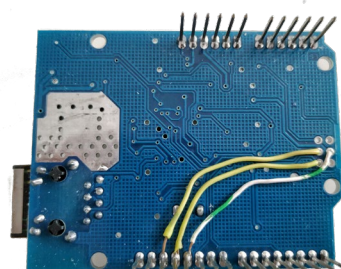



Ilustración 11. Tabla de conexión entre el conector ICSP del módulo Ethernet y la placa WEMOS (izquierda). Cableado realizado (derecha). Fuente propia, Abril 2019.

Se utiliza una caja específica para albergar una placa WEMOS junto con el módulo Ethernet, que cuenta con ranuras de fijación y aberturas realizadas o precortadas, por lo que a priori no se precisa realizar ninguna modificación de esta. Sin embargo, las dimensiones de la placa WEMOS son ligeramente superiores por lo que se han lijado sus lados más largos, así como las hendiduras de la caja para facilitar el acople de los componentes. Esta caja podrá albergar también el punto de acceso externo en caso de precisarse la utilización del mismo, tal y como se describe en el apartado 5.3.

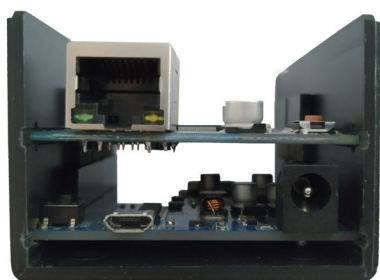


Ilustración 12. WEMOS D1 R1 con módulo Ethernet y caja DesignSpark. Fuente propia, Abril 2019.

### 5.2. Desarrollo hardware de las estaciones receptoras

El punto crítico de las estaciones receptoras son los indicadores luminosos, que informarán al operador de la utilización de su cámara durante una realización. Para adaptarse a las diferentes condiciones de

luz del entorno en el que se realiza la grabación, se requiere que dichos indicadores dispongan de una mínima intensidad luminosa y que esta pueda regularse.

Por otro lado, se quiere utilizar el adaptador USB de la cámara para alimentar el dispositivo. De este modo, se evita así el uso de baterías y/o la necesidad de una toma de corriente. De nuevo, se busca comodidad y rapidez para llevar a cabo el montaje del sistema de *tally*. En este mismo sentido, se determina la utilización de la zapata para flash externo de la cámara, como punto de anclaje del receptor.

Estas dos premisas, condicionarán fuertemente el diseño hardware. Los ledes que superan las 2 candelas de intensidad luminosa requieren, según sus especificaciones, de una corriente directa típica de 20mA y de una tensión directa máxima de 4V [18]. Sin embargo, la corriente nominal por pin de la NodeMcu es de 12mA. Para solventarlo, se utiliza un transistor lógico MOSFET (del inglés: *metal oxide semiconductor field effect transistor*), que permitirá proporcionar la corriente necesaria para la iluminación de los ledes. Esta corriente se consigue por medio del adaptador USB de la cámara, con el que alimentaremos el microcontrolador.

Con la premisa de poder conectar los ledes en cátodo común para facilitar su conexión externa, se propone el circuito que se muestra en la Ilustración 13. En este caso se ha seleccionado como transistor el BSS138, un MOSFET de enriquecimiento de canal N que actuará como interruptor. Al aplicar un nivel de tensión en el terminal gate o puerta ( $V_{GS}$ ) se controlará el estado de este, cerrado o abierto. Cuando  $V_{GS}$  sobrepase una tensión umbral ( $V_{th}$  típica del BSS138 = 1,3V), el transistor se encontrará en la zona de funcionamiento de saturación, actuando como un interruptor cerrado (véase ANEXO 7). De lo contrario, cuando no se aplique tensión,  $V_{GS}$  será 0, cumpliéndose que  $V_{GS} < V_{Th}$ . En este caso el transistor estará en la zona de funcionamiento de corte, donde la resistencia del transistor tiende a infinito, por lo que actuará como un circuito abierto, haciendo que la corriente circule por el diodo led.

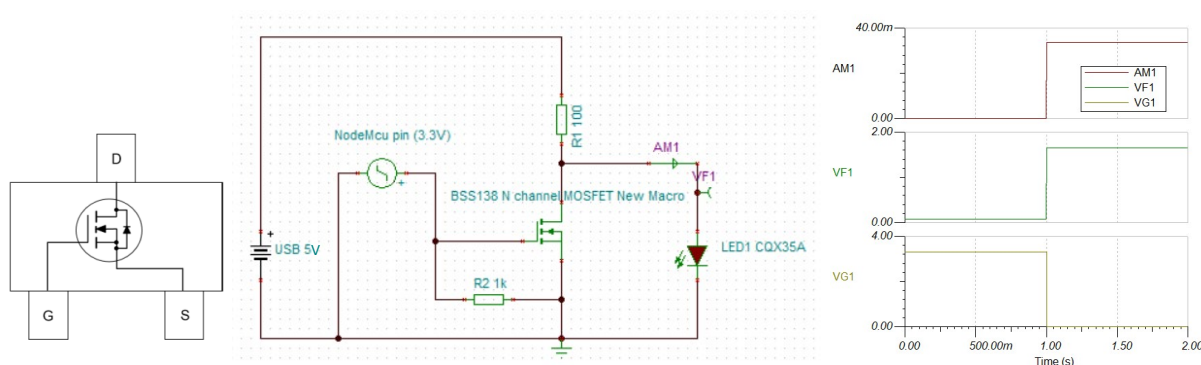


Ilustración 13. (Izquierda) BSS138 Datasheet. Fairchild Semiconductor, Octubre 2005. (Centro) Puerta lógica NOT, con un N-MOSFET. (Derecha) Análisis transitorio del circuito tipo realizado con Tina V9. Fuente propia, Mayo 2019.

De este modo, cuando no se aplique tensión sobre los pines de la NodeMcu que controlan el estado de los ledes (`digitalWrite(númeroPin, LOW);`), estos se iluminarán. Por el contrario, cuando se apliquen 3.3V (`digitalWrite(númeroPin, HIGH);`) los ledes permanecerán apagados.

Mediante la resistencia R1, se limita la corriente a 20mA marcada por el fabricante, con la que se obtendrían 20 candelas de intensidad luminosa en el led verde y 7,5 en el rojo. Teniendo en cuenta la tensión directa típica de cada led marcada por el fabricante, se calcula su valor para obtener la corriente deseada.

Resistencia para el led verde.	$\frac{5V-3,2V}{0,020 A} = 90\Omega \approx 100\Omega$
Resistencia para el led rojo.	$\frac{5V-2,1V}{0,020 A} = 145 \approx 150\Omega$

**Ecuación 1. Obtención del valor de las resistencias necesarias.**

Como puede observarse, se han escogido resistencias de la serie E12 cercanos a los valores calculados.

La resistencia R2 permite mantener apagado el transistor mientras no se establece un nivel lógico en el terminal de *gate*. Mientras el microcontrolador está arrancando, puede aparecer un retardo del orden de los milisegundos hasta que se establece un valor lógico en el puerto de salida. Los valores típicos suelen ser del orden de 1kΩ a 10 kΩ.

### 5.2.1. Desarrollo de la placa de circuito impreso (PCB)

Para conectar eléctricamente y sostener mecánicamente los componentes electrónicos del receptor, se realiza una placa de circuito impreso de dos capas conductoras. Las pistas serán de cobre y la base de resina de fibra de vidrio.

Se ha utilizado KiCad, en su versión 5.0.2, para realizar el diseño de la PCB. Se trata de un software con licencia GNU GPL para la automatización del diseño electrónico (del inglés: *Electronic Design Automation, EDA*). Este programa integra un editor de esquemáticos y un entorno de diseño de circuitos impresos con salida Gerber.

Para la posible utilización en un futuro de la salida de 3.3V de la placa de desarrollo para alimentar los ledes, en lugar de los 5V procedentes del adaptador USB de la cámara, se han añadido dos *jumpers* (JP1 Y JP2) que permite seleccionar la fuente deseada (véase Ilustración 14).

Se ha utilizado la configuración *pull up* (`pinMode(número_pin, INPUT_PULLUP)`) de las resistencias de los pines de entrada asociados al interruptor DIP para establecer el estado lógico *HIGH* para el estado de reposo de dichos pines. Por ello, el interruptor DIP está conectado directamente a los pines D1-D3 de la placa.



Con el diseño del esquemático, se plasma gráficamente el diseño descrito ya que representa una visualización de las interconexiones de los elementos del circuito a realizar. Para ello, se agregan los componentes a partir de las librerías instaladas o importadas en su defecto y se realizan las conexiones pertinentes. Tras realizar un control de reglas eléctricas, en el que KiCad revisa si hay inconsistencias en la relación entre las entradas y las salidas, se procede a asociar los componentes con sus huellas, se genera el *netlist* (archivo en el que se añaden las huellas de cada componente) y finalmente se ejecuta el editor de PCB para realizar el *layout*.

El *boardfile* es el archivo donde se realiza el diseño físico de la placa o *layout*. Como puede observarse en la Ilustración 14, el diseño de la PCB está limitado por el tamaño de la caja en la que irá ubicada. Por este motivo, se han hecho las mediciones pertinentes, considerando el tamaño y ubicación de los elementos que van fijados a la PCB. La ubicación de los conectores USB está limitada a uno de los laterales de la placa, para que esta coincida con la de su homólogo en la cámara. Así mismo, la placa de desarrollo deberá ir en la cara opuesta a los conectores USB.

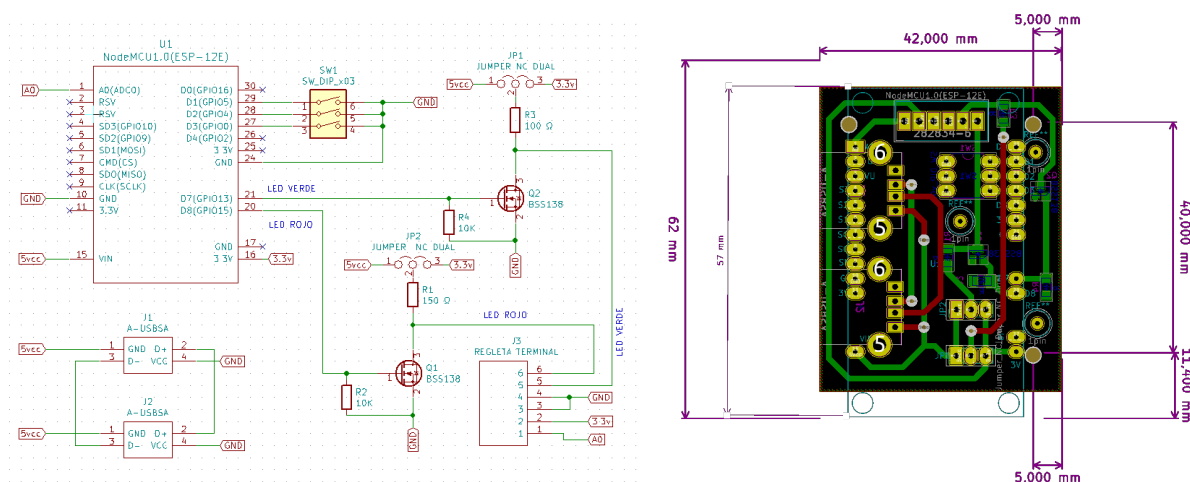


Ilustración 14. Esquemático (izquierda) y *layout* realizados con KiCad 5.0.2. Fuente propia, Abril 2019.

Teniendo en cuenta las premisas anteriores, se enrutan las pistas siguiendo el esquemático realizado e intentando minimizar la longitud de las pistas, reduciendo así los efectos parásitos resistivos y capacitivos. En algunos casos, ha sido necesario añadir una vía para dicho propósito. Se ha definido un plano de masa en cada una de las placas de cobre y, para mantener su continuidad, se han añadido vías interconectando ambos planos.

Se han diseñado tres agujeros en la PCB para fijar esta misma a la caja, mediante un tornillo de 2 mm diámetro exterior nominal. Para facilitar el soldado, se ha aumentado el diámetro de los pads (respetando el diámetro del orificio marcado por la huella) y se ha optado por formas rectangulares y

elípticas de estos. Con el mismo propósito, se ha aumentado el ancho de las pistas hasta 1 mm, pese a que 0,2mm sería suficiente para que circule una corriente de 0.5A.

Una vez diseñada la PCB, se han importado los archivos de los modelos 3D necesarios y se han asignado a las huellas correspondientes. Con ello, se podrá obtener la vista 3D que se muestra en la Ilustración 15 y que posteriormente se utilizará para el diseño del prototipo.

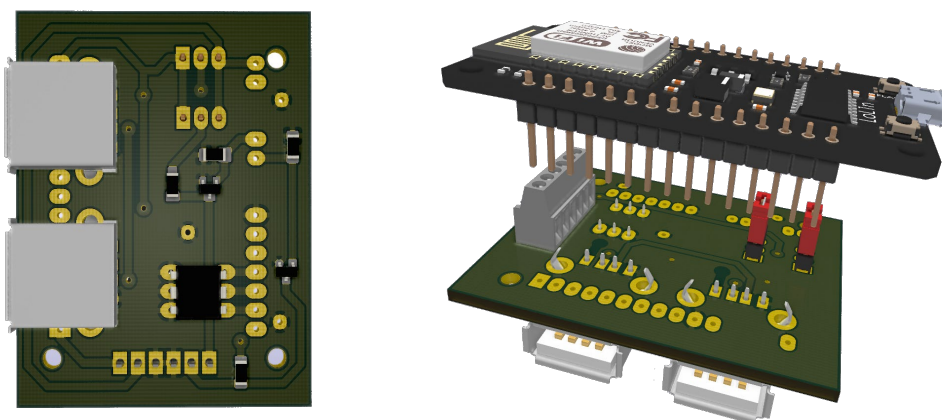


Ilustración 15. Vista 3D de la PCB generada con KiCad 5.0.2. Fuente propia, Abril 2019.

Finalmente se genera el archivo de formato Gerber para cada una de las capas, incluyendo la que marca el contorno de la placa. Este formato es un tipo de codificación de imágenes 2D con caracteres ASCII que representan las imágenes de las capas de cobre, los datos de perforación, enrutamiento y otros. Estos archivos, junto con el .drl generado, será procesado por el software de la fresadora o CNC (del inglés: *Computer Numeric Control*) para realizar el fresado de la PCB. Para ello, se ha utilizado el modelo ProtoMat S62 de la marca LPKF Laser & Electronics que incorpora el software Board Master.

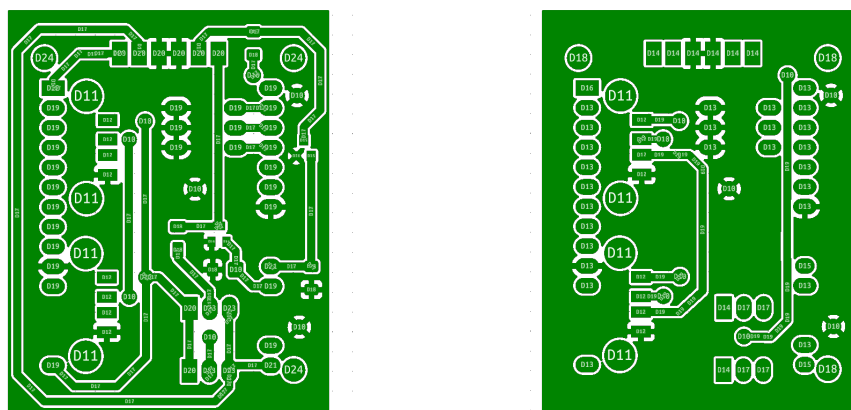


Ilustración 16. Vista de los archivos Gerber de las capas trasera (izquierda) y frontal (derecha) realizados con KiCad. Fuente propia, Abril 2019.

Tras el fresado, a la placa se le aplica una capa de recubrimiento, que previene la corrosión y las corrientes de fuga o cortocircuitos producto de la condensación. Una vez seca, se procede al soldado de los componentes que van fijados de manera directa a la placa con un soldador eléctrico y estaño. La placa de desarrollo se ensambla mecánicamente a una tira de zócalo y pines soldada a la PCB, elevando así su posición y dejando espacio para los *jumpers* y la regleta terminal. En esta última, se conectarán los ledes y el potenciómetro mediante cables de cobre, que permitirán la manipulación de la PCB una vez fijados los portaledes a la caja.

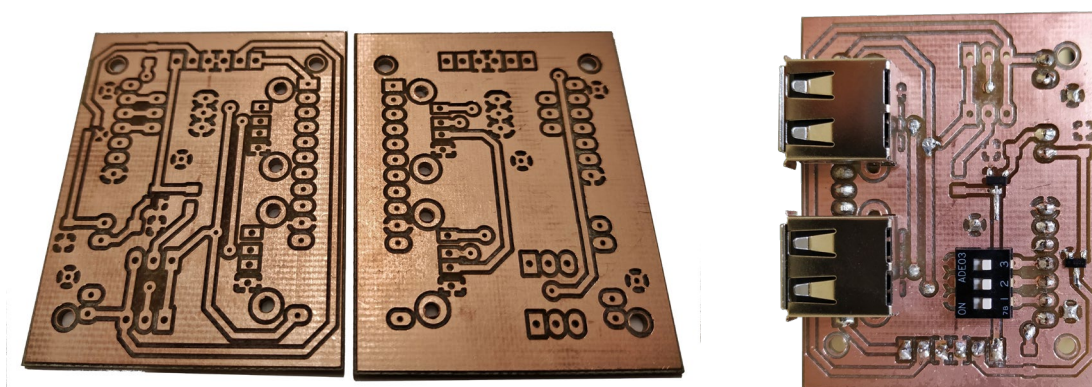


Ilustración 17. PCB realizada con la fresadora ProtoMat S62 (izquierda). Soldado de componentes a la PCB (derecha).  
Fuente propia, Abril 2019.

De manera simultánea al desarrollo de la PCB, y una vez se dispone de los componentes que integran el dispositivo receptor, se realiza el diseño del prototipo que se analiza en el próximo capítulo.

### 5.3. Diseño hardware de un punto de acceso externo

Tal y como se ha descrito en el punto 5.2.2, puede utilizarse un punto de acceso adicional para conseguir un tiempo de respuesta inferior y constante de los ledes. Para dicho propósito, se utiliza una placa de desarrollo NodeMcu Amica 1.0 V2 que se ubicará en la caja de la estación base y se alimentará por medio del pin de 5V del módulo Ethernet, que se conectará al pin Vin de la NodeMcu con un cable. También deberán conectarse los pines de masa de ambos.

En caso de querer aumentar la distancia entre el mezclador y los dispositivos receptores, sin necesidad de hacerlo mediante cableado Ethernet, podrá ubicarse la placa de desarrollo en una caja de plástico ABS Hammond 1551HGY (véase ANEXO 8). En este caso, la placa de desarrollo se alimentará mediante el conector micro-USB, a través de una fuente de alimentación de red con un adaptador AC/DC o bien con una batería externa.

## 6. Diseño del prototipo de las estaciones receptoras

Dadas las restricciones de espacio, se ha llevado a cabo el diseño de un modelo 3D del dispositivo receptor, para determinar la posición y encaje de cada uno de los elementos de este. En especial, es crítica la posición de los ledes y el potenciómetro, y sus respectivos embellecedores.

Para ello se ha utilizado el software de animación Autodesk MAYA 2018. Con este se han creado a escala real, la caja modelo TW / 5-4-7 de TAKACHI, según las especificaciones de la hoja técnica del producto (véase Anexo 2), así como el embellecedor del led. Por otro lado, se ha añadido a la escena la PCB y los componentes que van directamente soldados a esta, importándose el archivo 3D generado mediante el software KiCad. Por último, se han importado los archivos 3D del potenciómetro, que el fabricante pone a disposición.

Con todos los elementos en la escena y tras realizar las mediciones que pueden resultar críticas, se ha determinado la mejor ubicación de estos. Además, el modelo 3D permite saber con exactitud la posición de los orificios que hay que realizar en la caja, facilitando así este proceso. Gracias a este modelo se han descartado los planteamientos iniciales (véase la Ilustración 18), y se ha ubicado el potenciómetro en la parte superior. Esto, junto con la inclusión de unas arandelas en los porta ledes facilita el montaje del prototipo, así como su apertura y posterior manipulación de sus componentes.

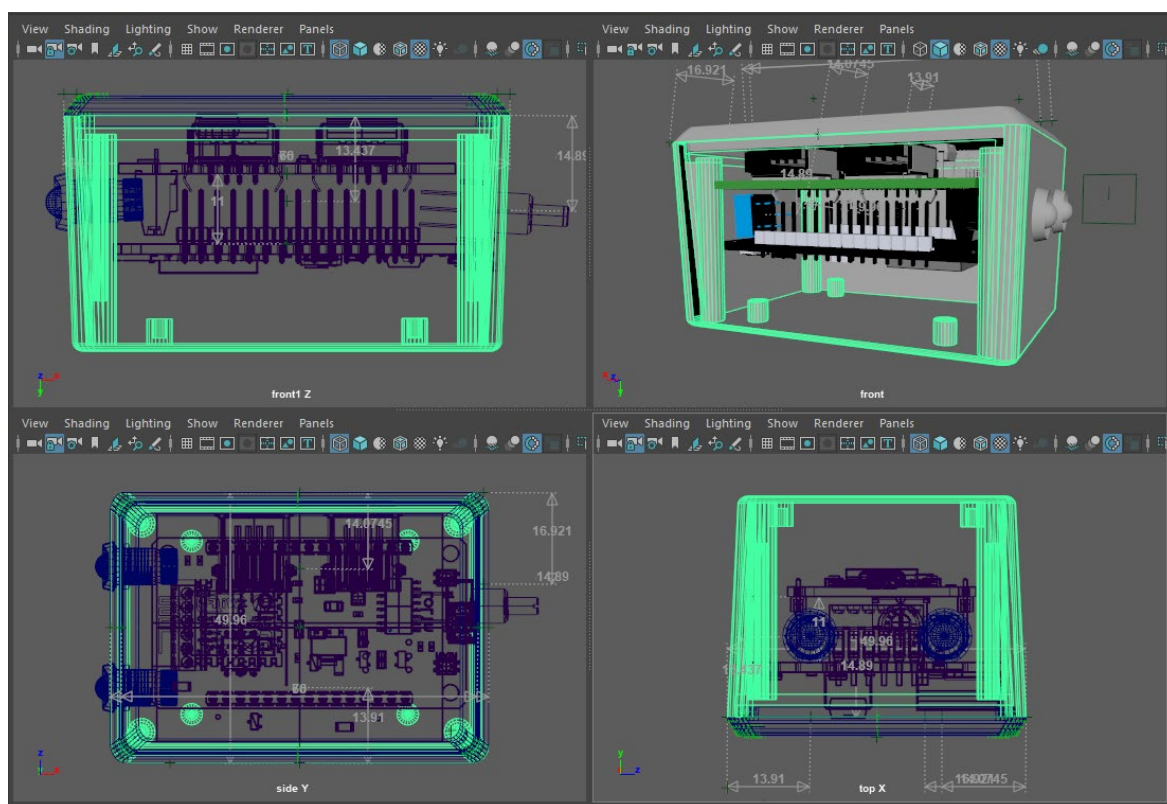


Ilustración 18. Diseño 3D del dispositivo receptor realizado con Maya 2018. Planteamiento inicial. Fuente propia, Abril 2019.



## 7. Conclusiones y trabajos futuros

Se han logrado cumplir los objetivos marcados al inicio del presente trabajo. Se ha realizado un sistema funcional basado en software y hardware libre que obtiene la información de *tally* de un mezclador ATEM y la envía, mediante tecnología inalámbrica, a los dispositivos receptores ubicados en las cámaras, que muestran dicha información en forma de indicador luminoso.

Se ha logrado adaptar la luminosidad de los ledes a las condiciones lumínicas del entorno de grabación. El tiempo de respuesta de los ledes obtenido, en especial con la utilización de un punto de acceso adicional, es adecuado para correcta funcionalidad del sistema.

Así mismo, el sistema es compatible con la versión software Blackmagic ATEM Switchers 7.5 instalada en los mezcladores de los que dispone el Grado en Ingeniería de Sistemas Audiovisuales de la UPC, impartido en la ESEIAAT. Por último, se ha logrado un alcance que supera con creces el marcado en los objetivos del presente trabajo.

A lo largo del proceso, se han superado las dificultades propias del mismo. No se tenía constancia que la placa Wemos carecía de conector ICSP, por lo que tras detectar los problemas de conexión con el módulo Ethernet, se procedió a realizar la conexión mediante cables de cobre.

El tamaño de caja de las estaciones receptoras ha limitado en exceso el diseño de la PCB y su posterior ensamblaje. Utilizar una caja de dimensiones superiores hubiese comportado menor complejidad al diseño.

Dada la poca documentación del ESP8266, no se ha deducido que el modo AP\_STA afecta a la comunicación Wi-Fi hasta que no se han realizado pruebas en una fase avanzada del proyecto. Con todo, se ha podido dar respuesta a dicha problemática, mejorando incluso las prestaciones del sistema (mayor alcance).

Para resolver las casuísticas en las que la conexión mediante Wi-Fi compromete el correcto funcionamiento del sistema, podría valorarse la opción de embeber la señal de tally con la señal de audio de los intercomunicadores. Este modelo no implicaría cableado adicional, no supondría mayor tiempo de montaje y aumentaría fiabilidad de la conexión, pudiéndose lograr el mismo alcance el propio sistema de *intercom*.

Se propone el desarrollo de una interfaz software para monitorizar el sistema. Este proporcionaría información del número de estaciones conectadas al punto de acceso, la calidad de las conexiones, el porcentaje de paquetes perdidos y de la pérdida de conexión con el mezclador, entre otros datos. De este modo se podrían tomar las medidas oportunas, mejorando la robustez del sistema. Pese a que el

sistema debe actuar en tiempo real, por lo que las rutinas no pueden aumentar en exceso la complejidad, estas acciones podrían gestionarse por interrupción, actuando solo en situaciones en las que la funcionalidad está o va a estar comprometida. Un servidor HTTP (del inglés: *Hypertext Transfer Protocol*) podría recibir información de estado de todas las estaciones, que se mostrarían en una interfaz realizada con HTML (del inglés: *HyperText Markup Language*). Esta también podría añadir funcionalidades para modificar parámetros de configuración de los dispositivos, como cambiar la IP del punto de acceso, el modo Wi-Fi, identificador de red, etc.

Por último, se propone estudiar el consumo de energía de los dispositivos y, si es necesario, implementar las mejoras oportunas para minimizarlo.

## 8. Presupuesto

A continuación, se muestra el coste de los materiales utilizados para la fabricación y proveer de alimentación a cada uno de los dispositivos que conforman el sistema de *tally* diseñado. En ambos casos, no se han incluido los materiales ni equipo necesario para el montaje de estos debido a que la finalidad del presente trabajo no es la realización de un producto comercial.

### 8.1. Presupuesto de la estación base

Vendedor	Código	Descripción	Cantidad	Precio(€) /unidad
RS Components España	144-2602	Arduino Uno Ethernet Shield Case - Black	1	8,25
e-ika	SKP: P1609	Shield Arduino UNO Ethernet W5100 R3	1	11,64
e-ika	SKP: P1012	Wemos D1 R1. Modulo WiFi ESP-12E D1, basado en ESP8266	1	10,11
RS Components España	737-8167	Fuente de alimentación Artesyn, 100 → 240V ac, 5V dc, 1 salida, 550mA, 3W	1	13,24
Amazon	-	UGREEN 11261 - Cable de Red (2 m, Cat7, U/FTP (STP), RJ-45, Negro)	1	7,99
TOTAL (€)				<b>51,23</b>

### 8.2. Presupuesto de una unidad de estación receptora

Vendedor	Código	Descripción	Cantidad	Precio(€) /unidad	Total
RS Components España	823-7367	Caja de ABS Takachi Electric Industrial TW5-4-7B, TW, No, 70 x 50 x 40mm	1	4,36	4,36
RS Components España	718-2073	Interruptor DIP, SPST, Pasante, Actuador Deslizante, 100 mA a 24 V d	1	0,6	0,6
RS Components España	756-1007	Regletas de terminales PCB, paso2.54mm 6 Contacto Hembra Recta Bloque	1	3,73	3,73
RS Components España	522-0625	Potenciómetro - Giraorrio, con 3,18 mm de diámetro , 10k	1	3,22	3,22
RS Components España	259-6812	Mando de potenciómetro RS PRO, eje 3.175mm, diámetro 11,6mm, Color Negro, indicador Blanco	1	1,05	1,05
RS Components España	674-1325	Zócalo USB tipo A ASSMANN WSW A-USBSA, Hembra, 1 puerto, Ángulo de 90°5	2	0,866	1,732
RS Components España	671-0324	MOSFET, BSS138	2	0,197	0,394
RS Components España	809-1638	LED Cree, Montaje en orificio pasante, Verde	1	0,231	0,231
RS Components España	810-6787	LED Cree, Montaje en orificio pasante, Rojo	1	0,166	0,166
RS Components España	237-0519	Indicador de soporte para bombilla LED Roscado	2	2,07	4,14
RS Components España	267-7416	Base múltiple Hembra Recta Winslow 2.54mm 32 pines 1 fila	1	1,45	1,45
RS Components España	164-8925	Resistencia fija TE Connectivity 100Ω	3	0,15	0,45
RS Components España	472-799	Resistencia fija TE Connectivity 10KΩ	6	0,256	1,536
RS Components España	224-0294	Resistencia fija TE Connectivity 150Ω	3	0,268	0,804
GNWE (Amazon)	-	SODIAL (R) 30 cm 1 pie USB 2.0 tipo A / A macho a macho	1	1,36	1,36
verwongee (Amazon)	-	Adaptador de Tripode 1/4 Pulgadas Tornillo a Soporte Zapata	1	0,88	0,88
e-ika	SKP: P0804	Modulo wifi ESP8266 - 12e NodeMcu, mod Lolin	1	8,87	8,87
TOTAL (€)					<b>34,973</b>

### 8.3. Presupuesto del punto de acceso externo

La tabla que se muestra a continuación muestra el conjunto de elementos necesarios para la fabricación y alimentación de un punto de acceso independiente de la estación base.

Vendedor	Código	Descripción	Cantidad	Precio(€) /unidad	Total
RS Components España	381-5091	Caja de ABS Hammond 1551HGY, 1551, IP54, 60 x 35 x 17mm	1	2,56	2,56
e-ika	SKP: P0804	Modulo wifi ESP8266 - 12e NodeMcu, mod Lolin	1	8,87	8,87
RS Components España	737-8167	Fuente de alimentación Artesyn, 100 → 240V ac, 5V dc, 1 salida, 550mA, 3W	1	13,24	13,24
TOTAL (€)					<b>24,67</b>





## 9. Bibliografía

- [1] Blackmagic Design, «ATEM Television Studio Switchers, Installation and Operation Manual,» June 2018. [En línea]. Available: <https://www.blackmagicdesign.com/>. [Último acceso: 27 02 2019].
- [2] K. Skaarhoj, «github,» SKAARHOJ ApS, [En línea]. Available: <https://github.com/kasperskaarhoj/SKAARHOJ-Open-Engineering>. [Último acceso: 03 2019].
- [3] A. Kalinchuk, «<https://github.com/kalinchuk>,» Kvitko Consulting, 2016. [En línea]. Available: [https://github.com/kalinchuk/ATEM\\_Wireless\\_Tally\\_Light](https://github.com/kalinchuk/ATEM_Wireless_Tally_Light). [Último acceso: 03 2019].
- [4] Cerevo Inc, «cerevo,» [En línea]. Available: <https://flectally.cerevo.com/en/docs/>. [Último acceso: 03 2019].
- [5] Cuebi, «Cuebi Light Manual,» [En línea]. Available: <https://www.cuebi.com/manual.html>. [Último acceso: 03 2019].
- [6] I. G. Gali, «Emitters and Receivers. Unit 4: Antennas,» UPC, Terrassa, 2018.
- [7] Espressif Inc, «ESP8266EX Datasheet,» 2018, p. 2.
- [8] The Abdus Salam International Centre for Theoretical Physics, «Cálculo del Presupuesto de potencia. Materiales de apoyo para entrenadores en redes inalámbricas,» 2011.
- [9] Arduino, «Arduino Duemilanove,» [En línea]. Available: <https://www.arduino.cc/en/Main/arduinoBoardDuemilanove>. [Último acceso: 03 2019].
- [10] I. Grokhotkov, «ESP8266WiFi library,» 2017. [En línea]. Available: <https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html>. [Último acceso: 05 2019].
- [11] M. Stör, «WiFi Module,» GitHub, 2019. [En línea]. Available: <https://github.com/nodemcu/nodemcu-firmware/blob/master/docs/modules/wifi.md>. [Último acceso: 05 2019].

- [12] J. M. J. L. M. O. E. Juanjo Alins, Multimedia content delivery in IP networks, Barcelona: Universitat Politècnica de Catalunya (UPC), 2019.
- [13] Espressif Inc, «ESP8266 Wi-Fi Channel Selection Guidelines, Application Note,» 2016, pp. 1-3.
- [14] e. a. Alexei V Nikitin, «Impulsive interference in communication channels and its mitigation by SPART and other nonlinear filters,» *EURASIP Journal on Advances in Signal Processing*, 2012.
- [15] Keysight Technologies, «IEEE 802.11 Wireless LAN, PHY Layer (RF) Operation and Measurement,» de *Application Note 1380-2*, 2017.
- [16] I. Grokhotkov, «Ticker.h,» 2014. [En línea]. Available: <https://github.com/esp8266/Arduino/blob/master/libraries/Ticker/Ticker.h>. [Último acceso: 04 2019].
- [17] Arduino, «Arduino Ethernet Shield Datasheet,» 2010. [En línea]. Available: [https://www.mouser.com/catalog/specsheets/A000056\\_DATASHEET.pdf](https://www.mouser.com/catalog/specsheets/A000056_DATASHEET.pdf). [Último acceso: 04 2019].
- [18] CREE, Inc, «CLD-CT1079.008: Cree 5-mm Red and Amber Round LED, PRODUCT FAMILY DATA SHEET,» 2019.

## ANEXO 1. Comparativa del Estado del Arte

Marca	Obtención señal	Traslado	Señalización	Compatibilidad con mezcladores	Ventajas	Inconvenientes	PVP para 4 receptores (€, Marzo 2019)
Blackmagic Design	ATEM Talkback Converter	Cable SDI	URSA Studio Viewfinder	ATEM Production Switchers	No añade más cableado Reliable	Precio Compatibilidad de inicio a fin	8951 <a href="https://www.blackmagicdesign.com/es">https://www.blackmagicdesign.com/es</a>
Blackmagic Design	ATEM Talkback Converter	Cable SDI	Blackmagic Studio Camera	ATEM Production Switchers	No añade más cableado Reliable Cámara incluida	Precio Compatibilidad de inicio a fin	7191 <a href="https://www.blackmagicdesign.com/es">https://www.blackmagicdesign.com/es</a>
Blackmagic + Datavideo	Blackmagic GPI / Tally interface + intercom ITC-100	Cable SDI	ITC-100SL + TD-2	ATEM Production Switchers	No añade más cableado Reliable	Precio Compatibilidad mezcladores	1870 <a href="https://www.blackmagicdesign.com/es">https://www.blackmagicdesign.com/es</a> <a href="https://www.datavideo.com/au/">https://www.datavideo.com/au/</a>
Blackmagic + Datavideo	Blackmagic GPI / Tally + cable DSUB25/15 + conversor TB-5	Cable TRS mini-jack	Luces TB-5	ATEM Production Switchers	Reliable	Compatibilidad mezcladores Tirada de cable extra Ciertos componentes descatalogados (stock y soporte limitado)	848,9 <a href="https://www.blackmagicdesign.com/es">https://www.blackmagicdesign.com/es</a> <a href="https://www.datavideo.com/au/">https://www.datavideo.com/au/</a>
SKAARHOJ	Tally Box System	Cable Ethernet	Tally lights	ATEM Production Switchers vMix TriCaster Mini	Reliable Compatibilidad múltiple Open source Pequeño y ligero	Tirada de cable extra	1299 <a href="https://www.skaarhoj.com/">https://www.skaarhoj.com/</a>
KVITKO	Transmisor Kviko	RF (433 / 915 Mhz)	Receptor Kvitko	ATEM Production Switchers	Inalámbrico Open source Pequeño y ligero	Posibles interferencias (not 100% reliable) Disponibilidad bajo demanda	1056,4 <a href="http://kvitko.com/tally/">http://kvitko.com/tally/</a>
CEREVO	Flexitally station (Ethernet/GPIO)	RF (315 / 433Mhz) Cable par paral-heap/entrelazado	Flexitally lamp	ATEM Production Switchers NewTek TriCaster Mini, 460, TC1 Roland VR-4HD SONY MCX-500 Panasonic AG-HMX100 DataVideo SE-1200M/U	Precio Compatibilidad múltiple Reliable (wired mode) Batería Pequeño y ligero	Posibles interferencias (wireless mode) Tirada de cable extra (wired mode)	467,64 <a href="https://flextally.cerevo.com/">https://flextally.cerevo.com/</a>
CUEBI	Punto de acceso + Cuebi Software WIFI (2.4Ghz)	WIFI (2.4Ghz)	Cuebi lights	ATEM Production Switchers NewTek TriCaster Roland Smart Tally vMix Ross Carbonite, Streamstar Any broadcast system supporting the TSL UMD v3.1 or v5.0 protocol.	Pequeño y ligero Compatibilidad múltiple Inalámbrico	Posibles interferencias (not 100% reliable)	599 <a href="https://www.cuebi.com/">https://www.cuebi.com/</a>

## ANEXO 2. Especificaciones Wi-Fi del ESP8266

Categories	Items	Parameters
Wi-Fi	Certification	Wi-Fi Alliance
	Protocols	802.11 b/g/n (HT20)
	Frequency Range	2.4G ~ 2.5G (2400M ~ 2483.5M)
	TX Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: -91 dbm (11 Mbps)
		802.11 g: -75 dbm (54 Mbps)
802.11 n: -72 dbm (MCS7)		
Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip	

Il·lustració 21. Especificacions tècniques del Wi-Fi del ESP8266. ESP8266EX Datasheet, Expressif Systems 2018.

### ANEXO 3. *Sketch* de las estaciones receptoras

```

#include <Ticker.h>
#include <ESP8266WiFi.h>
#include "user_interface.h" // Necesario para wifi_set_sleep_type(NONE_SLEEP_T);
#include <WiFiUdp.h>

#define LED_R D7 // RED LED
#define LED_G D8 // GREEN LED
#define T 50000 // Periodo PWM

bool led_state = false;
int duty=0; // Modificará el ciclo de Trabajo
int dipPins[] = {D3,D2,D1};
bool activatedR = false;
bool activatedG = false;
byte tally [] = {0,0,0,0,0,0,0,0,0};
int cam_adress;

void get_cam_adress(){ // Lee el estado del interruptor DIP y modifica el número de cámara
  int address;
  adress = (address << 1) | (!digitalRead(D3)); // Lee el valor del pin digital
  adress = (address << 1) | (!digitalRead(D2)); // Y desplaza el bit resultante a la izquierda
  adress = (address << 1) | (!digitalRead(D1));
  cam_adress = adress;
}

IPAddress MCastIP(239, 100, 1, 1); // Dirección multicast

WiFiUDP Udp; // Se inicializa el objeto WiDiUDP
unsigned int localUdpPort = 4211; // Puerto donde se recibirán los paquetes
const char* ssid = "ESPsoftAP_01"; // Identificador de la red
const char* password = "passT0p1!"; // Contraseña de la red Wi-Fi

void ICACHE_RAM_ATTR onTimerISR(){ // Función asociada a la rutina de interrupción
  led_state=!led_state;
  duty=analogRead(A0); // Lee el estado del potenciómetro
  if(led_state){ // Los ledes estarán apagados aquí
    timer1_write((duty*T)/1024); // Recalcula la siguiente interrupción
    digitalWrite(LED_G, HIGH);
    digitalWrite(LED_R, HIGH);
  }if(!led_state) { // Subsana comportamiento erróneo
    if(duty > 999){
      duty = 999;
    }
    timer1_write(((1024-duty)*T)/1024); // Recalcula la siguiente interrupción
    if(activatedR){ // En caso de estar en programa, se enciende el led rojo
      digitalWrite(LED_R, LOW);
    }
    if (activatedG){ // En caso de estar en anticipo, se enciende el led verde
      digitalWrite(LED_G,LOW);
    }
  }
}

void setup() {

  for(int i = 0; i<=2; i++){ // Se configuran los pines asociados al interruptor DIP
    pinMode(dipPins[i], INPUT_PULLUP); //Se configuran como entradas
    digitalWrite(dipPins[i], HIGH); // Se activan las resistencias internas asociadas
  }

  // Configuramos como salidas los pines de ledes y potenciómetro
  pinMode(LED_G,OUTPUT);
  pinMode(LED_R,OUTPUT);
  pinMode(A0,INPUT);

  // Inicializamos con los ledes apagados

```

```

digitalWrite(LED_G, HIGH);
digitalWrite(LED_R, HIGH);

// Configuramos conexión Wi-Fi
WiFi.mode(WIFI_STA); // Modo estación
wifi_set_sleep_type(NONE_SLEEP_T); // Subsana el problema de conexión con el AP
WiFi.begin(ssid, password); // Se inicia el intento de conexión
while (WiFi.status() != WL_CONNECTED){ // Esperamos hasta conectar con éxito
    delay(500);
    digitalWrite(LED_G, !(digitalRead(LED_G))); // Parpadeo de los ledes indica NO CONECTADO
    digitalWrite(LED_R, !(digitalRead(LED_R)));
}

Udp.beginMulticast(WiFi.localIP(),MCastIP,localUdpPort); // Inicia un socket UDP, en el puerto indicado

timer1_isr_init(); // Inicialización del Timer1 (16 bits)
timer1_attachInterrupt(onTimerISR); // Se implementa la rutina de interrupción
timer1_enable(TIM_DIV16, TIM_EDGE, TIM_SINGLE); // TIM_DIV16 => 5MHz (1677721.4 us max)
timer1_write(T); // Se llama a la rutina cada x ciclos de reloj

get_cam_adress(); // Obtenemos el número de cámara actual y inicializamos interrupciones
attachInterrupt(digitalPinToInterrupt(D1), get_cam_adress, CHANGE);
attachInterrupt(digitalPinToInterrupt(D2), get_cam_adress, CHANGE);
attachInterrupt(digitalPinToInterrupt(D3), get_cam_adress, CHANGE);
}

void loop(){

if (WiFi.status() != WL_CONNECTED){ // Si se pierde la conexión Wi-Fi, se intenta la reconexión
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED){
        delay(500);
        digitalWrite(LED_G, !(digitalRead(LED_G)));
        digitalWrite(LED_R, !(digitalRead(LED_R)));
    }
    Udp.beginMulticast(WiFi.localIP(),MCastIP,localUdpPort); // Iniciamos de nuevo el socket UDP
}

int packetSize = Udp.parsePacket(); //Devuelve el tamaño del paquete entrante(0 si no hay disponibles)

if (packetSize>0) {
    Udp.read(tally, sizeof(tally)); // Se almacenan los datos leídos en la variable tally
    if(tally[cam_adress]==0x1){ // Se modifican las variables que controlan el encendido de los ledes
        activatedR = true;
        activatedG = false;
    } else if (tally[cam_adress]==0x2){
        activatedG = true;
        activatedR = false;
    }else{
        activatedR = false;
        activatedG = false;
    }
}
}
}
}

```

## ANEXO 4. *Sketch* de la estación base sin punto de acceso propio

```

#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <EEPROM.h>
#include <SPI.h>
#include <ATEM.h>
#include <TextFinder.h>
#include <ATEMTally.h>
#include <Dhcp.h>
#include <Dns.h>
#include <Ethernet.h>
#include <EthernetClient.h>
#include <EthernetServer.h>
#include <EthernetUdp.h>
#include <WiFiUdp.h>

const char *ssid = "ESPsoftAP_01"; // Identificador de la red Wi-Fi
const char *password = "passT0p1!"; // Contraseña de la red Wi-Fi

IPAddress MCastIP(239, 100, 1, 1); // Dirección Multicast privada
WiFiUDP Udp;
unsigned int localUdpPort = 4210; // Puerto UDP local
unsigned int remotePort = 4211; // Puerto UDP remoto

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}; // Mac por defecto
byte ip[] = {192,168,10,234}; // Dirección IP por defecto

byte switcher_ip[] = {192,168,10,241}; // Dirección IP del mezclador ATEM
int switcher_port = 49910; // Puerto del mezclador ATEM

boolean doOnce = false; // Será true cuando se configure la conexión con el mezclador

ATEM AtemSwitcher; // Inicializamos objeto
ATEM
ATEMTally ATemptally; // Inicializamos objeto ATemptally

struct {
  int program; // Se define una estructura para la información de tally
  int preview;
} payload;

void setup(){

  WiFi.mode(WIFI_STA); // Modo estación
  WiFi.begin(ssid, password); // Se inicia el intento de conexión
  while (WiFi.status() == WL_DISCONNECTED){ // Esperamos hasta conectar con éxito
    delay(500);
  }

  Udp.beginMulticast(WiFi.localIP(),MCastIP, localUdpPort); // Iniciamos el socket UDP

  ATemptally.setup_ethernet(mac, ip, switcher_ip, switcher_port); // Inicia la conexión Ethernet con el ATEM

  delay(1000);

}

void loop(){

  if (!doOnce) { // Tan solo se ejecuta una vez
    // Se configura y posteriormente se intenta la conexión con el mezclador
    AtemSwitcher.begin(IPAddress(switcher_ip[0],switcher_ip[1],switcher_ip[2],switcher_ip[3]),switcher_port);
    AtemSwitcher.connect();
    doOnce = true;
  }

  if (WiFi.status() != WL_CONNECTED) { // Si se pierde la conexión Wi-Fi, se intenta la reconexión

```



```

WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED){
  delay(500);
}
Udp.beginMulticast(WiFi.localIP(),MCastIP, localUdpPort); // Iniciamos de nuevo el socket UDP
}

AtemSwitcher.runLoop();// Obtiene información del estado del mezclador y mantiene la conexión con este

if (AtemSwitcher.isConnectionTimedOut()) { // Se procede a reconectar si se ha perdido la conexión
  AtemSwitcher.connect();
} else {
  payload.program = AtemSwitcher.getProgramInput();
  payload.preview = AtemSwitcher.getPreviewInput();

  byte tally [] = {0,0,0,0,0,0,0,0,0,0}; // Inicializamos la variable tally

  if( payload.program == payload.preview){ // Modificamos el array tally
    tally[payload.program] = 0x1;
  }else{
    tally[payload.program] = 0x1;
    tally[payload.preview] = 0x2;
  }

  Udp.beginPacketMulticast(MCastIP, remotePort, WiFi.localIP()); // Inicia conexión UDP Multicast
  Udp.write(tally, sizeof(tally)); // Se escribe el datagrama UDP
  Udp.endPacket(); // Se envía el paquete UDP
}

delay(200); // Enviamos 5 paquetes por segundo
}

```

## ANEXO 5. Sketch del punto de acceso

```

#include <ESP8266WiFi.h>

const char *ssid = "ESPsoftAP_01"; // Identificador de la red Wi-Fi
const char *password = "passT0p1!"; // Contraseña de la red Wi-Fi
IPAddress APlocal_IP(192, 168, 10, 13); // IP del Punto de Acceso
IPAddress APgateway(192, 168, 10, 12); // Puerta de Enlace
IPAddress APsubnet(255, 255, 255, 0); // Máscara de Red

int scanResult (int networksFound){
  float rssi_array[16]= {-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150};
  float rssi_array2[16]={};
  float rssi, rssi_;
  float min_rssi = 999;
  int selection = -1;
  for (int i = 0; i < networksFound; i++){
    int pos = WiFi.channel(i)+1; // Posición de cada red encontrada en el array rssi
    rssi = WiFi.RSSI(i)/10.00; // dBm a mW paso 1
    float x = pow (10, rssi); // dBm a mW paso 2
    if (rssi_array[pos] == -150) {
      rssi_array[pos] = WiFi.RSSI(i);
    }else { // Suma fuentes incoherentes, sin correlación, con diferente offset de fase
      rssi_ = rssi_array[pos]/10.00;
      float y = pow(10, rssi_);
      rssi_array[pos] = 10 * (log10 (x+y)); // mW a dBm
    }
  }
  for (int j = 2; j < 14; j++){ // Se calculan las potencias recibidas de cada canal, añadiendo las interferencias
    // Caída de 12 dBm en la potencia de las señales dos canales a la izquierda
    float f1 = pow(10, (rssi_array[j-2]-12 / 10.00) );
    float f2 = pow(10, (rssi_array[j-1] / 10.00) );
    float f3 = pow(10, (rssi_array[j+1] / 10.00) );
    // Caída de 12 dBm en la potencia de la señal dos canales a derecha
    float f4 = pow(10, (rssi_array[j+2] -12 / 10.00) );
    float f0 = pow(10, (rssi_array[j] / 10.00) );
    rssi_array2[j] = 10 * ( log10 (f0 + f1 + f2 + f3 + f4) ); // Suma de fuentes incoherentes
  }
  for (int k = 2; k < 14; k++){ // Se busca el canal con la mínima potencia recibida
    if (rssi_array2[k] < min_rssi) {
      min_rssi = rssi_array2[k];
      selection = k-1;
    }
  }
  return selection;
}

void setup() {
  WiFi.scanNetworks();
  int canal; // El canal elegido. Se utilizará en WIFI.softAP(APssid, APpassword, canal)
  canal = scanResult(WiFi.scanComplete());
}

void setup() {
  Serial.begin(921600);
  WiFi.scanNetworks();
  // Llamada a la función para obtener el canal seleccionado
  int canal = scanResult(WiFi.scanComplete());
  WiFi.mode(WIFI_AP); // Modo Punto de Acceso
  Serial.println(WiFi.softAPConfig(APlocal_IP, APgateway, APsubnet) ? "OK" : "Failed!");
  Serial.println(WiFi.softAP(ssid, password, canal) ? "OK" : "Failed!"); // Establecimiento del PA
  Serial.print("Soft-AP IP address = ");
  Serial.println(WiFi.softAPIP()); // Se imprime la dirección IP del Punto de Acceso
}

void loop() { }

```

## ANEXO 6. Sketch de la estación base con punto de acceso propio

```

#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <EEPROM.h>
#include <SPI.h>
#include <ATEM.h>
#include <TextFinder.h>
#include <ATEMTally.h>
#include <Dhcp.h>
#include <Dns.h>
#include <Ethernet.h>
#include <EthernetClient.h>
#include <EthernetServer.h>
#include <EthernetUdp.h>
#include <WiFiUdp.h>

const char *ssid = "ESPsoftAP_01";           // Identificador de la red Wi-Fi
const char *password = "pasT0p1!";         // Contraseña de la red Wi-Fi

IPAddress APlocal_IP(192, 168, 10, 13);     // IP del Punto de Acceso
IPAddress APgateway(192, 168, 10, 12);     // Puerta de Enlace
IPAddress APsubnet(255, 255, 255, 0);     // Máscara de Red

IPAddress MCastIP(239, 100, 1, 1);        // Dirección Multicast privada
WiFiUDP Udp;
unsigned int localUdpPort = 4210;         // Puerto UDP local
unsigned int remotePort = 4211;          // Puerto UDP remoto

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}; // Mac por defecto
byte ip[] = {192,168,10,234};             // Dirección IP por defecto

byte switcher_ip[] = {192,168,10,241};    // Dirección IP del mezcladorATEM
int switcher_port = 49910;                //Puerto del mezclador ATEM

boolean doOnce = false;                   // Será true cuando se configure la conexión con el mezclador

ATEM AtemSwitcher;                        // Inicializamos objeto ATEM
ATEMTally ATemTally;                       // Inicializamos objeto ATemTally

struct {
  int program;                             // Se define una estructura para la información de tally
  int preview;
} payload;

int scanResult (int networksFound){        // Función para obtener el canal Wi-Fi adecuado

  float rssi_array[16]= {-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150,-150};
  float rssi_array2[16]={};
  float rssi, rssi_;
  float min_rssi = 999;
  int selection = -1;
  for (int i = 0; i < networksFound; i++){
    int pos = WiFi.channel(i)+1;           // Posición de cada red encontrada en el array rssi
    rssi = WiFi.RSSI(i)/10.00;             // dBm a mW paso 1
    float x = pow (10, rssi);              // dBm a mW paso 2
    if (rssi_array[pos] == -150) {
      rssi_array[pos] = WiFi.RSSI(i);
    }else { // Suma fuentes incoherentes, sin correlación, con diferente offset de fase
      rssi_ = rssi_array[pos]/10.00;
      float y = pow(10, rssi_);
      rssi_array[pos] = 10 * (log10 (x+y)) ; // mW a dBm
    }
  }
  for (int j = 2; j < 14; j++){ // Se calculan las potencias recibidas de cada canal, añadiendo las interferencias
    // Caída de 12 dBm en la potencia de las señales dos canales a la izquierda
    float f1 = pow(10, (rssi_array[j-2]-12 / 10.00) );
    float f2 = pow(10, (rssi_array[j-1] / 10.00) );
  }
}

```

```

float f3 = pow(10, (rssi_array[j+1] / 10.00) );
// Caída de 12 dBm en la potencia de la señal dos canales a derecha
float f4 = pow(10, (rssi_array[j+2] -12 / 10.00) );
float f0 = pow(10, (rssi_array[j] / 10.00) );
rssi_array2[j] = 10 * ( log10 (f0 + f1 + f2 + f3 + f4) ); // Suma de fuentes incoherentes
}
for (int k = 2; k < 14; k++){ // Se busca el canal con la mínima potencia recibida
    if (rssi_array2[k] < min_rssi) {
        min_rssi = rssi_array2[k];
        selection = k-1;
    }
}
return selection;
}

void setup(){

Serial.begin(921600);
Serial.println();
WiFi.scanNetworks();
// Llamada a la función para obtener el canal seleccionado
int canal = scanResult(WiFi.scanComplete());

WiFi.mode(WIFI_AP_STA); // Modo Punto de Acceso + Estación Wi-Fi
WiFi.softAPConfig(APlocal_IP, APgateway, APsubnet) // Configuración del Punto de Acceso
WiFi.softAP(ssid, password, canal); // Establecimiento del Punto de Acceso

delay(1000);

WiFi.begin(ssid, password); // Se inicia el intento de conexión
while (WiFi.status() == WL_DISCONNECTED){ // Esperamos hasta conectar con éxito
    delay(500);
}

Udp.beginMulticast(WiFi.localIP(),MCastIP, localUdpPort); // Iniciamos el socket UDP

ATEMTally.setup_ethernet(mac, ip, switcher_ip, switcher_port); // Inicia la conexión Ethernet con el ATEM

delay(1000);
}

void loop(){

if (!doOnce) { // Tan solo se ejecuta una vez
// Se configura y posteriormente se intenta la conexión con el mezclador
AtemSwitcher.begin(IPAddress(switcher_ip[0],switcher_ip[1],switcher_ip[2],switcher_ip[3]),switcher_port);
AtemSwitcher.connect();
doOnce = true;
}

if (WiFi.status() != WL_CONNECTED) { // Si se pierde la conexión Wi-Fi, se intenta la reconexión
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED){
    delay(500);
}
Udp.beginMulticast(WiFi.localIP(),MCastIP, localUdpPort); // Iniciamos de nuevo el socket UDP
}

AtemSwitcher.runLoop();// Obtiene información del estado del mezclador y mantiene la conexión con este

if (AtemSwitcher.isConnectionTimedOut()) { // Se procede a reconectar si se ha perdido la conexión
AtemSwitcher.connect();
} else {
    payload.program = AtemSwitcher.getProgramInput();
    payload.preview = AtemSwitcher.getPreviewInput();

byte tally [] = {0,0,0,0,0,0,0,0,0,0}; // Inicializamos la variable tally

if( payload.program == payload.preview){ // Modificamos el array tally

```

```
    tally[payload.program] = 0x1;
  }else{
    tally[payload.program] = 0x1;
    tally[payload.preview] = 0x2;
  }

  Udp.beginPacketMulticast(MCastIP, remotePort, WiFi.localIP()); // Inicia conexión UDP Multicast
  Udp.write(tally, sizeof(tally)); // Se escribe el datagrama UDP
  Udp.endPacket(); // Se envía el paquete UDP
}

delay(200); // Enviamos 5 paquetes por segundo
}
```

## ANEXO 7. Hoja de Características del BSS138



October 2005

BSS138

### BSS138

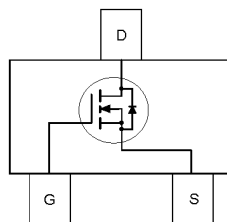
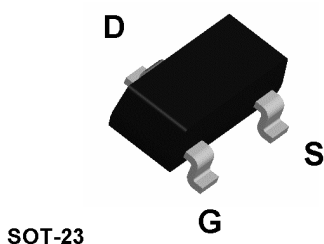
#### N-Channel Logic Level Enhancement Mode Field Effect Transistor

##### General Description

These N-Channel enhancement mode field effect transistors are produced using Fairchild's proprietary, high cell density, DMOS technology. These products have been designed to minimize on-state resistance while provide rugged, reliable, and fast switching performance. These products are particularly suited for low voltage, low current applications such as small servo motor control, power MOSFET gate drivers, and other switching applications.

##### Features

- 0.22 A, 50 V.  $R_{DS(ON)} = 3.5\Omega @ V_{GS} = 10 V$   
 $R_{DS(ON)} = 6.0\Omega @ V_{GS} = 4.5 V$
- High density cell design for extremely low  $R_{DS(ON)}$
- Rugged and Reliable
- Compact industry standard SOT-23 surface mount package



#### Absolute Maximum Ratings T<sub>A</sub>=25°C unless otherwise noted


Symbol	Parameter	Ratings	Units
V <sub>DSS</sub>	Drain-Source Voltage	50	V
V <sub>GSS</sub>	Gate-Source Voltage	±20	V
I <sub>D</sub>	Drain Current – Continuous (Note 1)	0.22	A
	– Pulsed	0.88	
P <sub>D</sub>	Maximum Power Dissipation (Note 1)	0.36	W
	Derate Above 25°C	2.8	mW/°C
T <sub>J</sub> , T <sub>STG</sub>	Operating and Storage Junction Temperature Range	-55 to +150	°C
T <sub>L</sub>	Maximum Lead Temperature for Soldering Purposes, 1/16" from Case for 10 Seconds	300	°C

#### Thermal Characteristics

R <sub>θJA</sub>	Thermal Resistance, Junction-to-Ambient (Note 1)	350	°C/W
------------------	--	-----	------

#### Package Marking and Ordering Information

Device Marking	Device	Reel Size	Tape width	Quantity
SS	BSS138	7"	8mm	3000 units

Electrical Characteristics <small>T<sub>A</sub> = 25°C unless otherwise noted</small>						
Symbol	Parameter	Test Conditions	Min	Typ	Max	Units
<b>Off Characteristics</b>						
BV <sub>DSS</sub>	Drain–Source Breakdown Voltage	V <sub>GS</sub> = 0 V, I <sub>D</sub> = 250 μA	50			V
$\frac{\Delta BV_{DSS}}{\Delta T_J}$	Breakdown Voltage Temperature Coefficient	I <sub>D</sub> = 250 μA, Referenced to 25°C		72		mV/°C
I <sub>DSS</sub>	Zero Gate Voltage Drain Current	V <sub>DS</sub> = 50 V, V <sub>GS</sub> = 0 V			0.5	μA
		V <sub>DS</sub> = 50 V, V <sub>GS</sub> = 0 V, T <sub>J</sub> = 125°C			5	μA
		V <sub>DS</sub> = 30 V, V <sub>GS</sub> = 0 V			100	nA
I <sub>GSS</sub>	Gate–Body Leakage.	V <sub>GS</sub> = ±20 V, V <sub>DS</sub> = 0 V			±100	nA
<b>On Characteristics</b> (Note 2)						
V <sub>GS(th)</sub>	Gate Threshold Voltage	V <sub>DS</sub> = V <sub>GS</sub> , I <sub>D</sub> = 1 mA	0.8	1.3	1.5	V
$\frac{\Delta V_{GS(th)}}{\Delta T_J}$	Gate Threshold Voltage Temperature Coefficient	I <sub>D</sub> = 1 mA, Referenced to 25°C		–2		mV/°C
R <sub>DS(on)</sub>	Static Drain–Source On–Resistance	V <sub>GS</sub> = 10 V, I <sub>D</sub> = 0.22 A V <sub>GS</sub> = 4.5 V, I <sub>D</sub> = 0.22 A V <sub>GS</sub> = 10 V, I <sub>D</sub> = 0.22 A, T <sub>J</sub> = 125°C		0.7 1.0 1.1	3.5 6.0 5.8	Ω
I <sub>D(on)</sub>	On–State Drain Current	V <sub>GS</sub> = 10 V, V <sub>DS</sub> = 5 V	0.2			A
g <sub>FS</sub>	Forward Transconductance	V <sub>DS</sub> = 10V, I <sub>D</sub> = 0.22 A	0.12	0.5		S
<b>Dynamic Characteristics</b>						
C <sub>iss</sub>	Input Capacitance	V <sub>DS</sub> = 25 V, V <sub>GS</sub> = 0 V, f = 1.0 MHz		27		pF
C <sub>oss</sub>	Output Capacitance			13		pF
C <sub>rss</sub>	Reverse Transfer Capacitance			6		pF
R <sub>G</sub>	Gate Resistance	V <sub>GS</sub> = 15 mV, f = 1.0 MHz		9		Ω
<b>Switching Characteristics</b> (Note 2)						
t <sub>d(on)</sub>	Turn–On Delay Time	V <sub>DD</sub> = 30 V, I <sub>D</sub> = 0.29 A, V <sub>GS</sub> = 10 V, R <sub>GEN</sub> = 6 Ω		2.5	5	ns
t <sub>r</sub>	Turn–On Rise Time			9	18	ns
t <sub>d(off)</sub>	Turn–Off Delay Time			20	36	ns
t <sub>f</sub>	Turn–Off Fall Time			7	14	ns
Q <sub>g</sub>	Total Gate Charge	V <sub>DS</sub> = 25 V, I <sub>D</sub> = 0.22 A, V <sub>GS</sub> = 10 V		1.7	2.4	nC
Q <sub>gs</sub>	Gate–Source Charge			0.1		nC
Q <sub>gd</sub>	Gate–Drain Charge			0.4		nC
<b>Drain–Source Diode Characteristics and Maximum Ratings</b>						
I <sub>S</sub>	Maximum Continuous Drain–Source Diode Forward Current				0.22	A
V <sub>SD</sub>	Drain–Source Diode Forward Voltage	V <sub>GS</sub> = 0 V, I <sub>S</sub> = 0.44 A (Note 2)		0.8	1.4	V
<b>Notes:</b>						
1. R <sub>θJA</sub> is the sum of the junction-to-case and case-to-ambient thermal resistance where the case thermal reference is defined as the solder mounting surface of the drain pins. R <sub>θJC</sub> is guaranteed by design while R <sub>θCA</sub> is determined by the user's board design.						
 <p>a) 350°C/W when mounted on a minimum pad..</p>						
Scale 1 : 1 on letter size paper						
2. Pulse Test: Pulse Width ≤ 300 μs, Duty Cycle ≤ 2.0%						

BSS138 Rev C(W)

## ANEXO 8. Hoja de características de la caja TWN 5-4-7

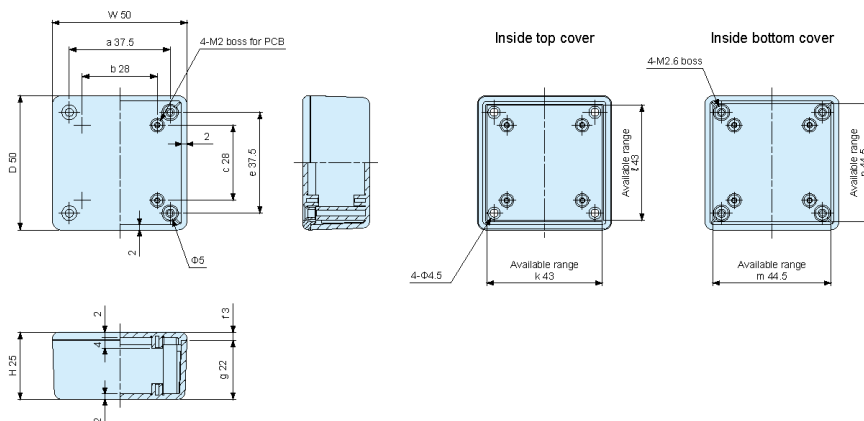


### TW · TWN Dimensions

• CAD data is downloadable from our web.

### TW·TWN SERIES

TW5-3-5

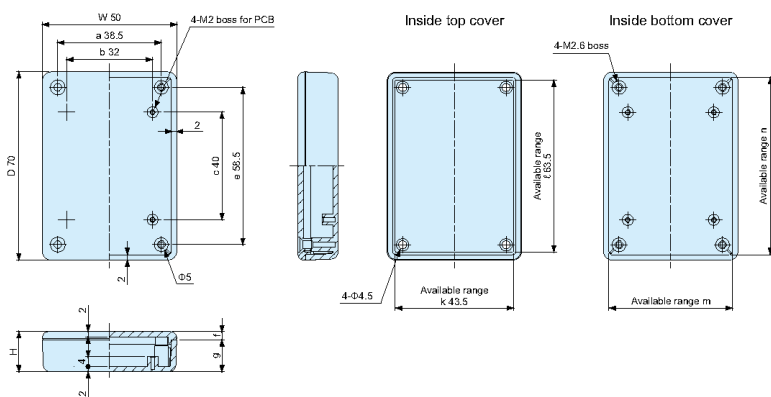


Part no.	W	H	D	a	b	c	e	f	g	i	j	k	ℓ	m	n
TW5-3-5□	50	25	50	37.5	28	28	37.5	3	22	—	—	43	43	44.5	44.5

TW / TWN5-2-7

TW / TWN5-3-7

TW / TWN5-4-7



Part no.	W	H	D	a	b	c	e	f	g	i	j	k	ℓ	m	n
TW / TWN5-2-7□	50	15	70	38.5	32	40	58.5	3	12	—	—	43.5	63.5	44.5	64.5
TW / TWN5-3-7□	50	27.5	70	38.5	32	40	58.5	3	24.5	—	—	43.5	63.5	44	64
TW / TWN5-4-7□	50	40	70	38.5	32	40	58.5	3	37	—	—	43.5	63.5	43	63



## ANEXO 9. Diagrama de Gant

