ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Sistema de transmisión de imágenes sobre plataforma de bajo consumo y largo alcance

(System of image transmission on low power, long range platform)

Para acceder al Título de

Graduado en

Ingeniería de Tecnologías de Telecomunicación

Autor: Miguel Ruiz García

Diciembre – 2016

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Miguel Ruiz García
Director del TFG: Pablo Sánchez Espeso
Título: "Sistema de transmisión de imágenes sobre plataforma de bajo consumo y largo alcance".
Title: "System of image transmission on low power, long range platform"
Presentado a examen el día: 23/12/2016

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Pablo Sánchez Espeso
Secretario (Apellidos, Nombre): Francisco José Alcalá Galán
Vocal (Apellidos, Nombre): Víctor Fernández Solórzano

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente	Fdo.: El Secretario
Fdo.: El Vocal	Fdo.: El Director del TFG (sólo si es distinto del Secretario)
V° B° del Subdirector	Trabajo Fin de Grado N° (a asignar por Secretaría)

Agradecimientos:

Siento la obligación de acordarme , como no podía ser de otra manera, de mis padres y mi hermano porque gracias a ellos soy lo que soy, además de cuidarme y ayudarme siempre que estoy en apuros (y cuando no también).

A mis amigos que han conseguido que en estos meses de duro trabajo no haya perdido (del todo) el contacto con el mundo real y que cada rato que estoy con ellos es como si nada cambiara.

Por último, gracias a mi tutor Pablo por guiarme en este proyecto, así como de escuchar hasta la más absurda de mis ocurrencias.

Índice:

1 INTRODUCCIÓN

1.1 CONTEXTO

Estando el auge de la era del "Internet de las Cosas" (de ahora en adelante IoT, "Internet of Things") por llegar, es de esperar que millones de componentes que utilizamos cotidianamente (neveras, televisiones, cámaras) estén conectados a internet. Se permite de esta manera al usuario tener la capacidad de controlar y chequear el estado de cada uno de los dispositivos de su hogar desde cualquier lugar.

A raíz de esta revolución cobran especial importancia tecnologías inalámbricas como Lora [10], que poseen características que las hacen ideales en aplicaciones relacionadas con IoT y comunicaciones máquina a máquina (M2M). La tecnología Lora ofrece la posibilidad de trabajar a largas distancias (del orden de Km), con un bajo consumo y protección frente a interferencias, proporcionando alta cobertura en lugares cerrados. Gracias a estas características es posible implementar una infraestructura de comunicaciones de tipo LPWAN (Low Power Wide Area Network) y que además sea una solución de bajo coste y escalable. Desde el punto de vista tecnológico, Lora proporciona receptores con una alta sensibilidad, lo que hace que no sea necesario transmitir con potencias altas (<20dBm).

"Internet de las Cosas" es un mercado en el que se prevé una dura batalla entre empresas dedicadas típicamente al desarrollo software (como Google) y otras (como Robert Bosch GmbH o ST Microelectronics) que orientan su negocio a la fabricación de componentes. Las grandes inversiones que se están realizando en este área durante los últimos años dan pistas de hacia dónde se va a mover el mercado.

Según un informe de Tata Consultancy Services [1] sobre 715 empresas en EEUU, el aumento medio de ingresos después de invertir en tecnología Lora aumentó un 16,5%. De hecho, las expectativas de inversión en iniciativas basadas en IoT experimentarán un significativo aumento en el período 2015-2018 (como muestra la Figura 1), que será superado en la etapa 2018-2025.

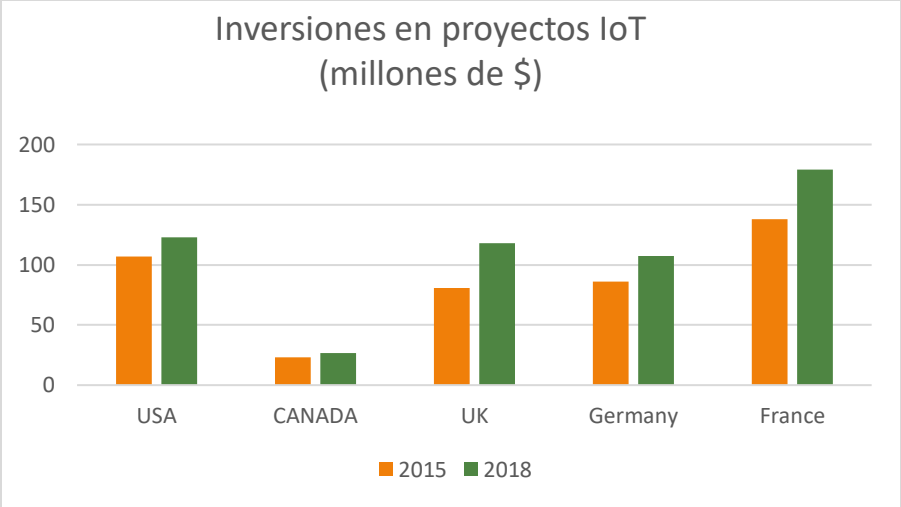


Figura 1. Inversión media de las empresas en los respectivos países.

La tendencia positiva en este ámbito es clara, por lo que las empresas ven una buena oportunidad de negocio y de potenciales clientes.

Otro dato que se incluye en el estudio mencionado, es la mediana de las inversiones en este tipo de proyectos. Dicho parámetro se obtiene colocando en orden ascendente el capital puesto en juego y seleccionando la cantidad intermedia en esta sucesión para hacer la media aritmética entre estos valores.

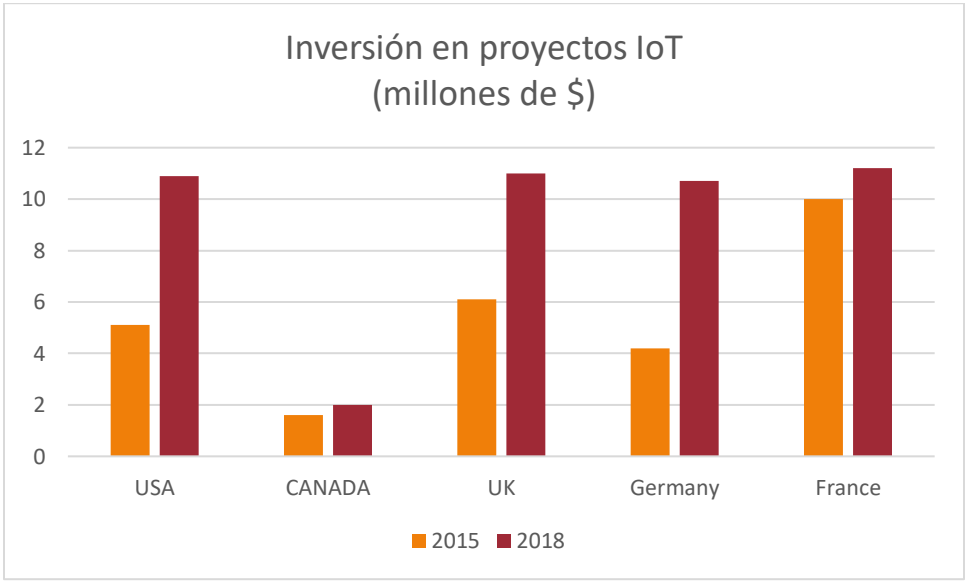


Figura 2. Mediana de las inversiones en millones de $. Fuente: TCS (Tata Consultancy Services).

Las cantidades invertidas son, en términos absolutos, mucho menores que en el caso de la inversión media, por lo que se constata que gran parte del desarrollo de I+D se está llevando a cabo en grandes corporaciones, las cuales tienen mayor capacidad de destinar recursos a

investigación y desarrollo de nuevos proyectos. Por otra parte, cabe reseñar que empresas más pequeñas, a pesar de invertir menos (como es lógico), aumentarán sus inversiones porcentualmente, de manera espectacular en países como EEUU, el Reino Unido o Alemania.

1.2 MOTIVACIÓN

Como se ha comentado en el punto anterior, la tecnología Lora tiene un enorme atractivo tanto a nivel de desarrollo tecnológico como de inmediata aplicación industrial. Sin embargo, una de las limitaciones de dicha tecnología es que se orienta a aplicaciones en las que la tasa de transmisión es baja, es decir, se va a utilizar típicamente para controlar una alarma anti-incendios, posibles malfuncionamientos de electrodomésticos, etc. En este tipo de aplicaciones, el dispositivo se va a encontrar normalmente en un estado de bajo consumo, sin actividad ni acceso a red.

Este proyecto, en vez de plantear una aplicación relacionada con la domótica (seguramente el mayor mercado potencial de Lora), desarrolla un sistema que envía imágenes con baja tasa de transmisión. Esta aplicación permite profundizar en conceptos de compresión y procesado de imágenes así como analizar el consumo de este tipo de dispositivos cuando se varía la tasa de transmisión de bits en perjuicio del alcance máximo.

Es también un reto el hecho de poder desarrollar desde cero un compresor JPEG y ver cómo fluctúa la tasa de compresión cambiando los distintos agentes que lo forman. Reafirmando así los conocimientos teóricos adquiridos durante el Grado en temas de procesado de imagen y vídeo.

1.3 ESPECIFICACIÓN

El objetivo del proyecto es desarrollar un sistema que comprima una imagen, utilizando el estándar JPEG, y lo transmita utilizando la tecnología Lora. El desarrollo se realizará íntegramente en C y se iniciará en un PC de sobremesa. Una vez validado en un PC, el desarrollo será portado a una plataforma embebida. Se ha seleccionado como plataforma embebida la

tarjeta "STM32 Nucleo Board" [14]. Debido a limitaciones de memoria en la plataforma embebida, la imagen se tendrá que enviar por ráfagas desde el PC a la plataforma embebida, que se encargará de procesarla y transmitirla. Por lo tanto, la imagen será capturada en el PC, el cual irá transmitiendo (a través del puerto serie) la porción de la imagen que corresponda a la tarjeta "Nucleo Board". El sistema embebido, además de comprimir la imagen y gestionar la comunicación el PC, se encarga de manejar el transceptor Lora y de establecer un protocolo de comunicaciones para que el transmisor y receptor Lora sean capaces de entenderse. Dicho protocolo debe ser capaz de gestionar errores en la comprobación de CRC, pérdida de paquetes y otros problemas de comunicación, de manera que el transmisor sepa en todo momento si un paquete ha sido recibido correctamente y, si no es así, proceder a su retransmisión.

A continuación se describen las especificaciones que debe cumplir el diseño final:

1. El sistema debe ser capaz de comprimir una imagen en el menor tiempo posible y transmitirla dentro de un entorno en el que las distancias no superen la decena de kilómetros.
2. Ante la imposibilidad de guardar una imagen VGA completa (resolución 640x480 pixeles, con una profundidad de pixel de 24 bits con formato RGB), la aplicación debe estar preparada para hacer un procesado de la imagen por ráfagas y transmitirla de igual manera.
3. Es necesario desarrollar una aplicación de PC que vaya transmitiendo al sistema embebido la información necesaria de la imagen en cada momento.
4. Definir e implementar un protocolo que permita transmitir las imágenes de manera que se asegure todo lo posible la recepción de información, evitando la perdida de imágenes.
5. Chequear la funcionalidad de la comunicación a largas distancias (hasta 10 Km).

Cabe destacar que la tecnología inalámbrica en la cual se basan los transceptores sustenta su largo alcance en su capacidad de recuperar señales de muy baja intensidad. Si principal virtud es la de proporcionar receptores de optimas prestaciones (muy alta sensibilidad) a muy bajo coste.

2 ESTADO DEL ARTE

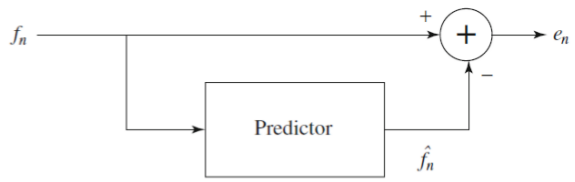
Este capítulo se divide en tres partes. En primer lugar se analiza el algoritmo de compresión JPEG. A continuación se estudiarán aspectos relacionados con la red de comunicaciones y el protocolo Lora (secciones 2.2 y 2.3). Por último, en la tercera parte (sección 2.4) se presentan los elementos hardware que conformarán el sistema a diseñar.

2.1 COMPRESIÓN DE IMAGEN

Son numerosos los algoritmos capaces de reducir el tamaño en disco necesario para almacenar una foto, tales como JPEG, PNG o RAW. Inicialmente dichos algoritmos se clasifican en algoritmos con pérdidas o sin pérdidas. Si bien un procedimiento sin pérdidas (Lossless) tiene la ventaja de que la imagen será recuperada sin ninguna distorsión, los factores de compresión conseguidos no son lo suficientemente satisfactorios como para utilizarlos en la mayoría de aplicaciones reales. Como contrapartida, los algoritmos con pérdidas proporcionan factores de compresión más altos, lo que permite un ahorro de espacio en disco y un mayor aprovechamiento del ancho de banda. Es de obligación advertir que se producirá una pérdida de calidad de la imagen que será directamente proporcional al nivel de compresión alcanzado.

2.1.1 Compresión sin pérdidas

Los algoritmos sin perdidas suelen utilizar un codificador predictivo que reduce las redundancias espaciales entre pixeles contiguos. Este predictor puede ser de muchos tipos, siendo el más simple aquel que selecciona como predicción una función que tiene en cuenta el valor de los pixeles anteriores (por ejemplo el valor medio). El resultado obtenido del predictor es normalmente sustraído al valor real, y la diferencia enviada a un codificador de entropía (típicamente de tipo Huffman). Como es bien sabido, debido a la redundancia espacial de la imagen las diferencias entre pixeles contiguos son normalmente pequeñas, lo que reduce el número de bits en la codificación.



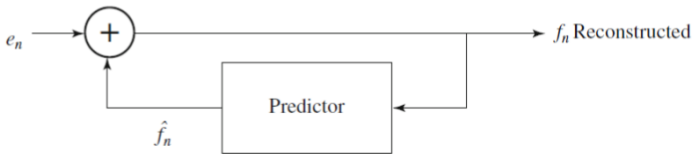


Figura 6. Esquema de un codificador y decodificador predictivo sin pérdidas.

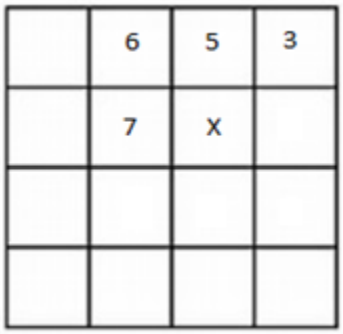
A medida que se va introduciendo un nuevo pixel, representado por f_n, se va a calcular su predicción (f'_n) en base a entradas anteriores y este resultado se resta al valor original. Es de esta manera cómo únicamente se codifica el error [8]:

$$e_n = f_n - f'_n$$

Al otro lado, el decodificador realiza la operación inversa, obteniendo la muestra reconstruida:

$$\tilde{f}_n = e_n + f'_n$$

Como se ha comentado anteriormente, una de las funciones de predicción más sencilla es la que calcula la media entre los valores adyacentes.



$$X = \frac{5+6+7+3}{4} = 5$$

Figura 7. Ejemplo de codificador predictivo.

2.1.2 Compresión con pérdidas

El principal inconveniente de utilizar una compresión sin pérdidas (lossless) es que no se alcanzan unos ratios de compresión lo suficientemente elevados para aplicaciones multimedia. Esta es la causa por la que se emplean algoritmos con pérdidas, en los que la señal decodificada no se corresponde exactamente con la original, viéndose mermada su calidad pero reduciendose el espacio necesario para almacenarla.

Como principales diferencias a nivel estructural entre un compresor con pérdidas y sin perdidas se pueden citar la introducción de un cuantificador (principal causa de la distorsión de la imagen) y la sustitución del predictor por una transformada, como puede ser la de Fourier, el coseno o la transformada de Wavelet. El cuantificador reduce el rango de valores de salida haciendo que haya una menor dependencia entre coeficientes y propiciando el

truncamiento de valores al cero. Este es uno de los factores clave de la compresión, ya que a mayor cantidad de coeficientes eliminados, el ratio de compresión aumentará en la misma medida.

Particularmente, se ha utilizado en este proyecto la compresión JPEG y por lo que este algoritmo que se va a explicar brevemente a continuación.

2.1.2.1 Algoritmo de compresión JPEG

Como se ha comentado, JPEG introduce pérdidas en la imagen y reduce la redundancia espacial mediante la transformada discreta del coseno (DCT de ahora en adelante). El algoritmo incluye 6 pasos:

1. Transformación del espacio de color de RGB a YCbCr.
2. Sub-muestreo de la imagen.
3. Transformada del coseno.
4. Cuantización.
5. Ordenación en zig-zag de los coeficientes cuantizados.
6. Codificación de los coeficientes.

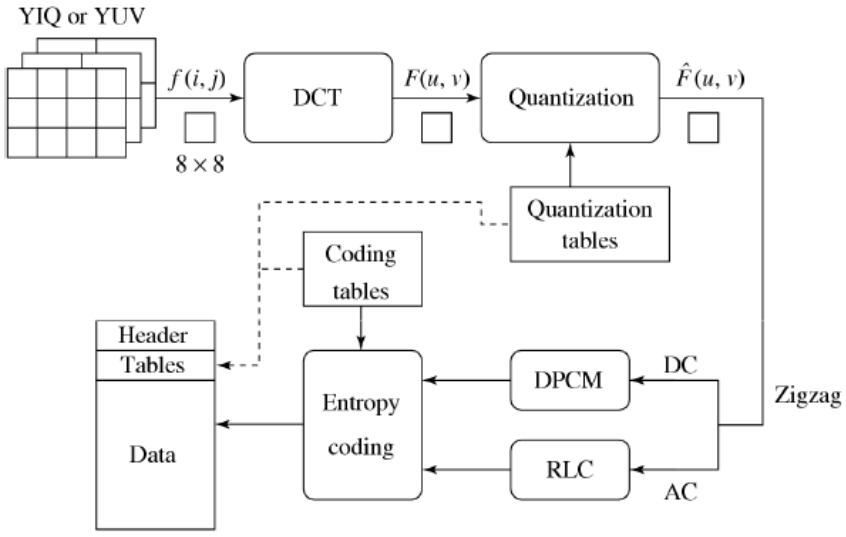


Figura 8. Diagrama de bloques de un compresor JPEG.

1. La transformación del espacio de color tiene en cuenta que el ojo humano es más sensible a la luminancia que a la crominancia. Es decir, el ser humano es capaz de discernir con mayor facilidad un cambio en el tono de una escala de grises que en el color de una foto. Al separar por un lado el canal con la luminancia y por otro la diferencia de color se posibilita una mejora de la compresión con un bajo impacto en la calidad de la imagen. El modelo RGB (Red, Green, Blue) es un modelo de color basado en la síntesis, por la que se posibilita la representación de cualquier color en función de

la mezcla de los tres canales disponibles: rojo, verde y azul. Cada uno de ellos tendrá un valor comprendido entre 0 (mínima intensidad, negro) y 255 (máxima intensidad), por lo que puede ser implementado con 8 bits.

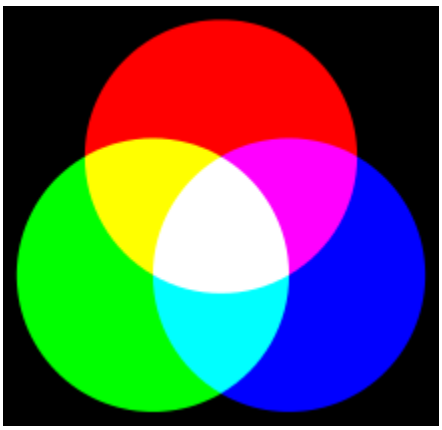


Figura 9. Modelo de color aditivo RGB.

Este modelo de uso de color implica redundancia en la imagen ya que la información relativa al brillo de la imagen está triplicada (aparece en los tres canales). Para ser capaces de aprovechar esta característica es necesario cambiar el modelo RGB a otro que no incluya este tipo de redundancias. Por ello se utiliza el modelo YCbCr. Este modelo representa cada pixel mediante 3 componentes: la luma o luminancia (Y) y la croma o crominancia (Cb,Cr), siendo Cb la diferencia de azul y Cr la diferencia de rojo-. Este es un estándar similar al YUV pero con un diferente escalado de croma.

$$C_b = ((B' - Y')/1.772) + 0.5$$
$$C_r = ((R' - Y')/1.402) + 0.5$$

Figura 10. Escalado de croma azul y roja [8].

Como se puede observar Cb representa la diferencia escalada entre la componente azul y la luminancia. Cr se calcula de forma similar, pero con la componente roja.

2. El sub-muestreo de croma permite reducir el número de compontes de color que se utilizan en la compresión. El submustreo es consecuencia de la baja sensibilidad del ojo humano a las variaciones de color. Uno de los muestreos más comunes es 4:2:2.

3. La transformada discreta del coseno permite representar una secuencia de datos como una suma de varias ondas coseno a distintas frecuencias, siendo ampliamente utilizada en algoritmos de compresión de imagen/video. Hay que aclarar que el procesado se realizará sobre pequeñas matrices de 8x8 pixeles, denominadas macrobloques, que se irán tratando independientemente. Esto provoca el típico efecto mosaico en imágenes con alto ratio de compresión, dado que la imagen no se trata como un todo, sino macrobloque a macrobloque. La expresión de la transformada DCT en 2 dimensiones para un macrobloque de tamaño 8x8 es la siguiente:

$$F(u,v) = \frac{C(u)\,C(v)}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} f(i,j)$$

i, j, u, v: 0, 1, …,7

$$C(\xi) = \begin{cases} \frac{\sqrt{2}}{2} & if \quad \xi = 0, \\ 1 & otherwise. \end{cases}$$

No obstante, para visualizar el efecto en un macrobloque es mucho más gráfico el ejemplo de la figura adjunta, en donde se muestran los pesos que se multiplican por cada pixel del macrobloque en cada coeficiente de la transformada. Los pesos abarcan desde el blanco (1) hasta el negro (-1).

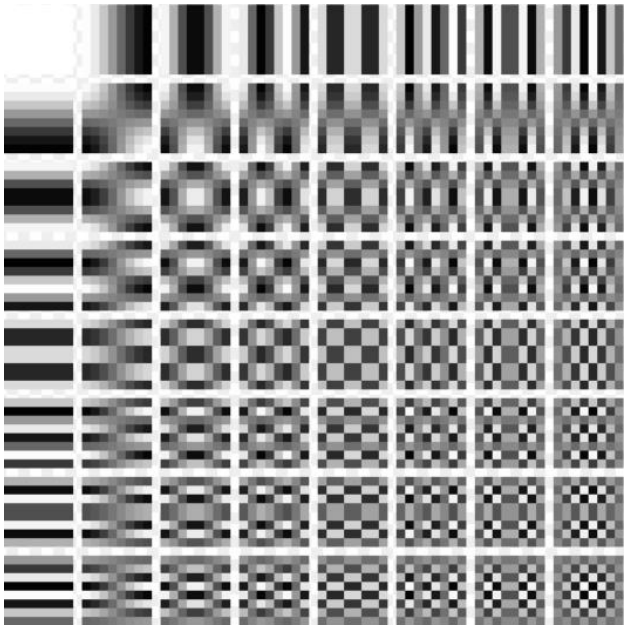


Figura 12.Función 8X8 2D transformada del coseno

En la parte superior derecha se encuentran las componentes frecuenciales bajas (la frecuencia espacial da cuenta de la cantidad de veces que cambian un pixel de color en una zona) y a medida que se hace un desplazamiento hacia parte inferior derecha se visualizan las componentes de alta frecuencia. En función de la similitud de una imagen con un patrón concreto de los que están presentes en la figura 10, esa componente tendrá mayor peso o no. En el siguiente ejemplo se muestra como en una imagen absolutamente en blanco, la única componente no nula es la componente (0,0) de continua o "Direct Current" (DC). Todas las demás componentes (coeficientes de alterna o AC, "Alternating Current") son cero.

```
a=[255 255 255 255 255 255 255 255;
   255 255 255 255 255 255 255 255;
   255 255 255 255 255 255 255 255;
   255 255 255 255 255 255 255 255;
   255 255 255 255 255 255 255 255;
   255 255 255 255 255 255 255 255;
   255 255 255 255 255 255 255 255;
   255 255 255 255 255 255 255 255]

 x= dct(a,-1)
```

```
x  =

   2040.    0.    0.    0.    0.    0.    0.    0.
      0.    0.    0.    0.    0.    0.    0.    0.
      0.    0.    0.    0.    0.    0.    0.    0.
      0.    0.    0.    0.    0.    0.    0.    0.
      0.    0.    0.    0.    0.    0.    0.    0.
      0.    0.    0.    0.    0.    0.    0.    0.
      0.    0.    0.    0.    0.    0.    0.    0.
      0.    0.    0.    0.    0.    0.    0.    0.
```

Figura 13. Ejemplo de la transformada del coseno para una imagen en blanco.

En el siguiente ejemplo se va a introducir una imagen con rayas blancas y negras horizontales. Ya se puede adelantar que las principales componentes serán las de la primera columna.

```
a=[0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0;
   255 255 255 255 255 255 255 255;
   255 255 255 255 255 255 255 255;
   0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0;
   255 255 255 255 255 255 255 255;
   255 255 255 255 255 255 255 255]

 x= dct(a,-1)
```

```
x  =

   1020.         0.    0.    0.    0.    0.    0.    0.
 - 382.83688     0.    0.    0.    0.    0.    0.    0.
      0.         0.    0.    0.    0.    0.    0.    0.
 - 783.54131     0.    0.    0.    0.    0.    0.    0.
      0.         0.    0.    0.    0.    0.    0.    0.
   523.54557     0.    0.    0.    0.    0.    0.    0.
      0.         0.    0.    0.    0.    0.    0.    0.
   76.150991     0.    0.    0.    0.    0.    0.    0.
```

Figura 14. Ejemplo de la transformada del coseno para una imagen a rayas horizontales blancas y negras.

A simple vista se podía intuir que la posición (0,3) iba a ser la de mayor valor (en valor absoluto) ya que esta es la posición cuyo patrón se asemeja más a la imagen.

4. El proceso siguiente es el de cuantización. Es este el principal introductor de error (distorsión) en el sistema en el que trata de reducir el rango de valores de salida. Las tablas de cuantización tienden a tener valores altos hacia la esquina inferior derecha para reducir el peso de las componentes de alta frecuencia de la DCT. Los valores resultantes de esta transformación serán divididos término a término por el correspondiente número de la tabla. Si bien hay numerosas tablas que se pueden utilizar, a partir de los recomendados en la bibliografía [4] se han utilizado las siguientes:

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Figura 15. Tabla de cuantificación de luminancia

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Figura 16. Tabla de cuantificación de la crominancia.

Se puede percibir cómo se reafirma el concepto introducido inicialmente en el que se expone cómo el ojo humano es más sensible a la escala de grises que al color en el hecho de que los valores en la tabla de la crominancia son más uniformes y tienden rápidamente hacia un valor constante.

Lo normal es que los valores obtenidos de la cuantización tiendan a cero a medida que se realiza un desplazamiento en el macrobloque hacia abajo y a la derecha.

5. El siguiente paso es hacer una ordenación en zig-zag de los coeficientes cuantizados, de modo que los coeficientes de valor cero queden lo más agrupados posible en las últimas posiciones de un vector. Esta es la clave final de la compresión, ya que va a permitir transmitir una gran cantidad de información con muy pocos bits, utilizando codificaciones de tipo RLC (Run Length Coding).

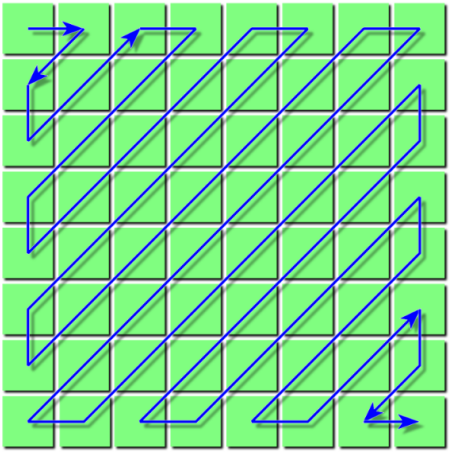


Figura 17. Ordenación en zig-zag de un macrobloque 8x8.

6. La ordenación en Zig-Zag permite la conversión de una matriz de 8x8 a un vector de 64 elementos. A continuación se pasa a realizar una codificación de tipo RLE (Run Length Encoding), para los coeficientes AC. Este algoritmo codifica la información por pares correspondientes a la cantidad de veces que aparece un símbolo consecutivamente y al propio símbolo. A continuación se muestra un ejemplo de cómo trabaja la compresión RLE.

Mensaje: XXXYABBBBBB5554 Codificación: 3X1Y1A6B3514

Figura 18. Codificación RLE

$$RC = \frac{Longitud\_mensaje}{Longitud\_codificación} = \frac{15}{12} = 1{,}25$$

Bien es cierto que en entornos donde haya una generación de símbolos aleatoria, la repetición de estos puede ser considerablemente improbable y provocar que la compresión sea negativa, en otras palabras, el archivo comprimido tenga un tamaño mayor que el original. Sin embargo, gracias al procesado previo de imagen esto no va a ocurrir. Para los coeficientes de DC -el primer término de la transformada, (0,0)- se realiza una codificación basada en DPCM (Difference Pulse Coding Modulation), por el que se manda el primer término sin modificar y, a partir de ese momento, se procede a enviar la diferencia entre el coeficiente DC actual y el anterior, como se muestra en la figura.

X= 69 65 65 70 67 DPCM= 69 -4 0 5 -3

Figura 19. Ejemplo de codificación DPCM.

Por último, a las cantidades resultantes obtenidas se les aplica una codificación de entropía, típicamente un código Huffman. El objetivo es asignar a los valores que se repiten más (valores con mayor probabilidad de aparición) códigos más cortos que a los de menor probabilidad, reduciendo de esta manera la longitud media del código y

acercándolo al límite que marca la entropía. Se define la entropía como la cantidad media de información por símbolo generada por una fuente. Su expresión teórica aparece en la figura adjunta.

$$H(s) = \sum_{i=1}^{q} p(si) \cdot \log_2\left(\frac{1}{p(s_i)}\right) = \sum_{i=1}^{q} p(si) \cdot I(s_i)$$

$$I(s_i) = \log_2\left(\frac{1}{p(s_i)}\right)$$

Figura 20. Definición de entropía.

Se define la longitud media del mensaje como:

$$L = \sum_{i=1}^{q} l_i \cdot p(s_i)$$

Figura 21. Definición de longitud media

Donde li es la longitud de la palabra Si. Cuanto más se acerque este valor a la entropía mayor será la eficiencia del código. Dicha eficiencia oscila entre cero y uno, como muestra la siguiente figura.

$$\eta = \frac{H(x)}{L} \quad 0 < \eta \leq 1$$

Figura 22. Definición de eficiencia en la generación de un código.

El algoritmo para obtener un código Huffman ordena todos los símbolos en una fila con probabilidad descendente. El algoritmo incluye 2 pasos que permiten generar el código:

1. Agrupar los dos símbolos con menos probabilidad e iniciar el árbol con esos dos símbolos para crear un nodo padre.
2. Se añade otro símbolo a la lista original con la probabilidad de la suma de los dos símbolos agrupados anteriormente y se repite el proceso.

2.2 INFRAESTRUCTURA DE REDES LPWAN

Dadas las actuales tendencias en el mercado de las telecomunicaciones, es de esperar un tremendo aumento de los dispositivos conectados a internet. Según Forrester [9], dicha

cantidad ascenderá a los 22 billones (americanos) para el año 2020. Esto supondrá un aumento del tráfico que no podrá ser absorbido sin tener que duplicar o incluso triplicar las estaciones GSM instaladas. Es por esto que las redes LPWAN tienen un potencial incremento espectacular.

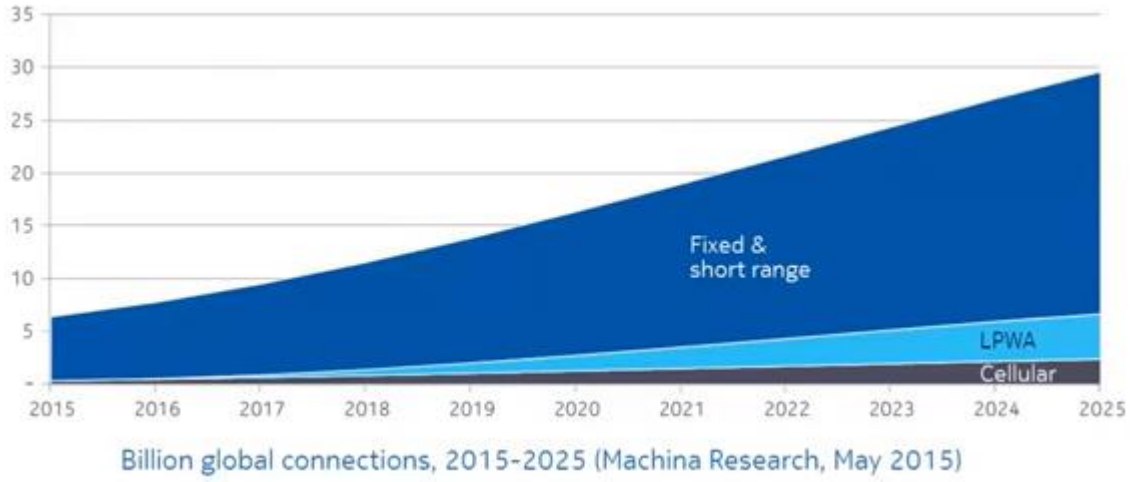


Figura 23. Evolución de las redes de comunicaciones en el período 2015-2025.[13]

Como se muestra en la gráfica, las comunicaciones M2M están dominadas por las aplicaciones de radio cortas; redes propietarias en su mayoría que funcionan sin operador, típicamente utilizadas para sistemas de alarma. El interés por aumentar su radio de acción ha provocado que se deban colocar numerosos repetidores para no perder la señal, aumentando por ello los costes. Las LPWAN se sitúan como principal candidato para resolver esta limitación, debido a su facilidad para operar con dispositivos alimentados por batería, de bajo consumo, baja frecuencia de comunicación y velocidad de transmisión.

Estas redes pueden utilizar una modulación Lora, que facilita un mayor rango de actuación que otras tecnologías. Esta modulación está basada en técnicas de espectro ensanchado (Spread Spectrum), lo cual incrementa la seguridad frente a interferencias y ruido. La topología de una red LPWAN se distribuye según una topología en estrella en el que hay tres actores principales: dispositivo final, puerta de acceso (Gateway), servidor.

Los gateways se comportan como meros repetidores en la topología en estrella que reciben la señal de RF procedente de un cierto sensor y lo retransmiten al servidor central vía IP.

Este proyecto se centra en la comunicación entre un módulo transmisor y receptor basados en modulación Lora, por lo que se está testando esa primera conexión dispositivo final – puerta de enlace. Los receptores son capaces de demodular varias señales sobre distintos canales simultáneamente. De hecho son capaces de recuperar una señal 19,5dB por debajo del nivel de ruido (u otra señal interferente). A modo de comparación, sistemas que utilizan FSK necesitan

señales entre 8-10dB por encima del nivel de ruido para poder demodular la señal. Los receptores de Lora son capaces de atender 8 frecuencias distintas con 2 factores de ensanchado (spreading factor, SF) distintos para cada una de ellas. En caso de colisión, la señal con mayor potencia será decodificada. [12]

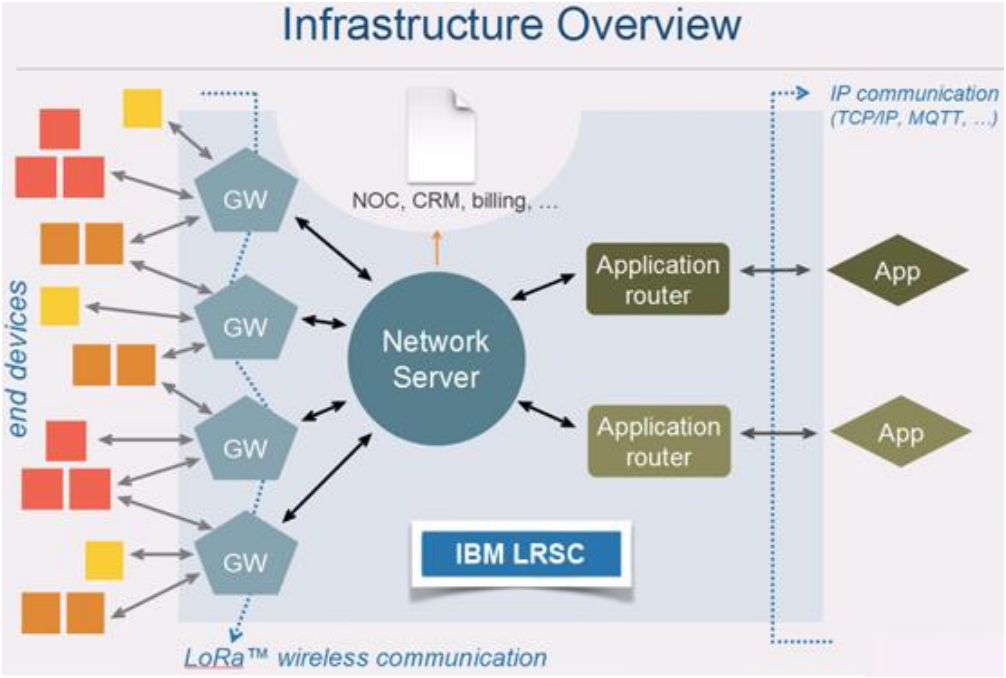


Figura 24. Esquema de una red de comunicaciones LPWAN [9]

Es conveniente recalcar que es posible en estos sistemas una comunicación bidireccional, facilitando la transmisión desde el servidor central hasta la puerta de enlace, y de esta a los sensores. Un ejemplo de este tipo de comunicación la constituye la transmisión, desde el servidor central, de actualizaciones del software de los sensores y/o puertas de enlace.

El espectro de frecuencia usado por este tipo de dispositivos va desde los 137MHz hasta los 1020MHz. En el caso del transceptor SX1276 incluye dos antenas optimizadas para funcionar a las frecuencias de 433 MHz y de 868 MHz. En cuanto a la tasa de transmisión de datos (data rate) la misma va a estar limitada por la distancia a la que deba enviarse el mensaje y la duración del mismo.

Para aumentar el “data rate” y/o que la duración de la transmisión se reduzca es necesario aumentar el ancho de banda, lo cual repercute negativamente en la sensibilidad del receptor. Esto hace que puede que el número de paquetes que lleguen con error o se pierdan sea demasiado elevado, haciéndolo de esta manera inviable el sistema de comunicación.

Por esta razón, en este tipo de sistemas es también necesario que las puertas de enlace (gateways) puedan cambiar su velocidad de transmisión para poder asegurar la fiabilidad del

enlace. En otras palabras, los nodos más cercanos al Gateway utilizarán una mayor velocidad de transmisión y una menor potencia de salida, mientras que los que se encuentran en el límite del alcance (link budget) tratarán de optimizar la sensibilidad del receptor disminuyendo la velocidad de transmisión y maximizando la potencia. Este sistema ha recibido el nombre de ADR (Adaptative Data Rate).[10]

Aunque en este proyecto se haga especial énfasis en la tecnología Lora, existen otras alternativas como SigFox o LTE-M. SigFox y Lora tienen una visión distinta del negocio del IoT que se puede resumir de la siguiente manera [16]:

- SigFox planea establecerse como un estándar que se comporte como un operador a nivel global de IoT.
- Lora es proveedora de una tecnología que permite a otras empresas desarrollar sus aplicaciones IoT de forma independiente.

Ambas mantienen un mismo espíritu de implementar comunicaciones de muy bajo consumo, reducido tráfico y velocidad de transmisión, así como un alto rango de operación en base a dispositivos con muy alta sensibilidad. La denominada SigFox Cloud sirve de intermediaria entre el cliente y el dispositivo proveedor del servicio, aunque hay que tener en cuenta que la red de comunicaciones de SigFox debe cubrir el área de trabajo en el que se mueva el usuario.

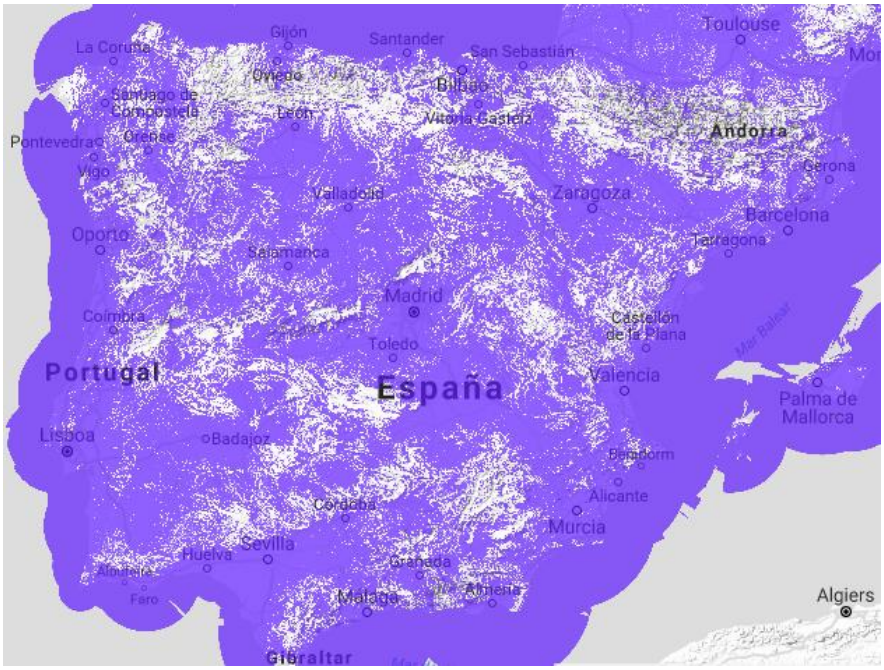


Figura 25. Cobertura de la red de comunicaciones IoT de SigFox.[15]

A continuación se comparan las características de los dispositivos Lora y de SigFox.

Característica	Lora	SigFox
Max. Output power	20 dBm	14 dBm
Max. Link budget	168 dB	161
Max. sensitivity	-148 dB	-147 dB
Frequency	433/868MHz	865-870MHz
Máximo alcance	~15Km	~15Km
Tx consumption	50mA(a 14 dBm)	60mA (a 14dBm)
Sleep consumption	1µA	7µA
Rx consumption	12mA	35mA
Bit rate	0.3-22kbps	100bps

Se observa cómo la diferencia en cuanto a sensibilidad y distancia máxima de funcionamiento es prácticamente nula, pero SigFox presenta claras desventajas en cuanto a consumo y principalmente en el "bit rate", el cual está limitado a 100bps. Además, la cantidad de mensajes diarios que se pueden transmitir son de 140, por lo cual se vislumbra una gran desventaja con respecto a Lora.

2.3 Modulación Lora

Las redes LPWAN no tienen a nivel tecnológico nada que ver con la modulación Lora, ya que se podrían implementar utilizando cualquier otra modulación típica (por ejemplo FSK).

Lora utiliza técnicas de espectro ensanchado, como Chirp Spread Spectrum (CSS), por lo que utiliza un ancho de banda mayor que el estrictamente necesario para transmitir la información que se desea enviar. Como es de especial interés incrementar el rango de distancia al que se quiere transmitir, se debe aumentar la energía por bit. Esto se puede hacer aumentando la potencia de transmisión, lo cual no es una opción ya que la regulación de la UE (Unión Europea) limita a 14 dBm la potencia máxima con la que se puede transmitir en la banda de 868 Mhz. Otra opción es utilizar una nueva modulación.

Un "chirp" es una señal sinusoidal cuya frecuencia va a variar linealmente con el tiempo, como se muestra en la figura adjunta [4].

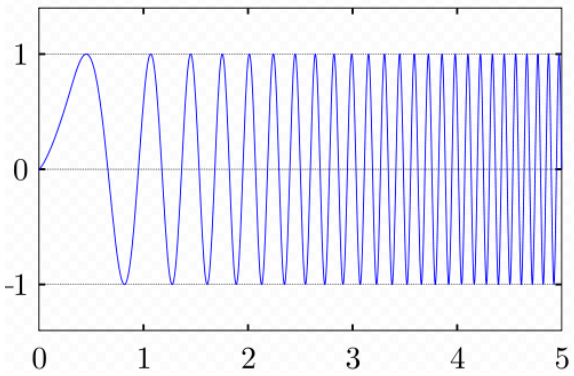


Figura 2.1. Esquema de una red de comunicaciones LPWAN

Un espectrograma es una representación en tres dimensiones de una señal, que incluye información temporal, frecuencial y de potencia de la señal (ver figura adjunta). Si se utilizase un espectrograma para representar la señal se obtendría la siguiente imagen (suprimiendo la potencia de la señal) [13]..

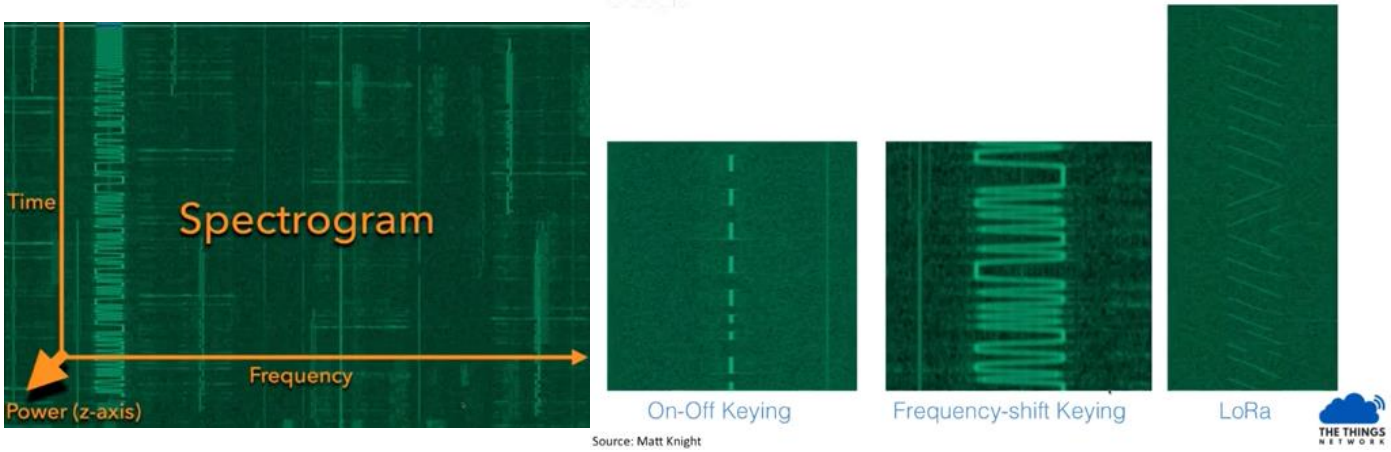


Figura 2.2. Esquema de una red de comunicaciones LPWAN

A la izquierda se presenta la forma más simple de transmitir: On-Off Keying (si hay que transmitir un bit a '1' se transmite una señal de una determinada frecuencia y si es '0' no se trasmite nada). Otra de las modulaciones más empleada es la FSK (Frecuency-Shift keying), por la cual la información es transmitida en base a cambios de frecuencia discretos de una señal portadora.

En cuanto a la modulación Lora, se puede observar cómo cada la frecuencia de cada chirp cambia linealmente con el tiempo. La diferencia entre la frecuencia más baja y la más alta proporciona la información de la codificación de cada símbolo. Si bien se aprecia cómo inicialmente hay uniformidad en la señal, luego, a pesar de recibir el mismo tipo de señales hay pequeños saltos. Este desplazamiento de la fase se corresponde a la información de cada símbolo.

Para ver en más detalle cómo funciona esta modulación se va a utilizar como herramienta la siguiente figura, con un ejemplo de demodulación de señal.

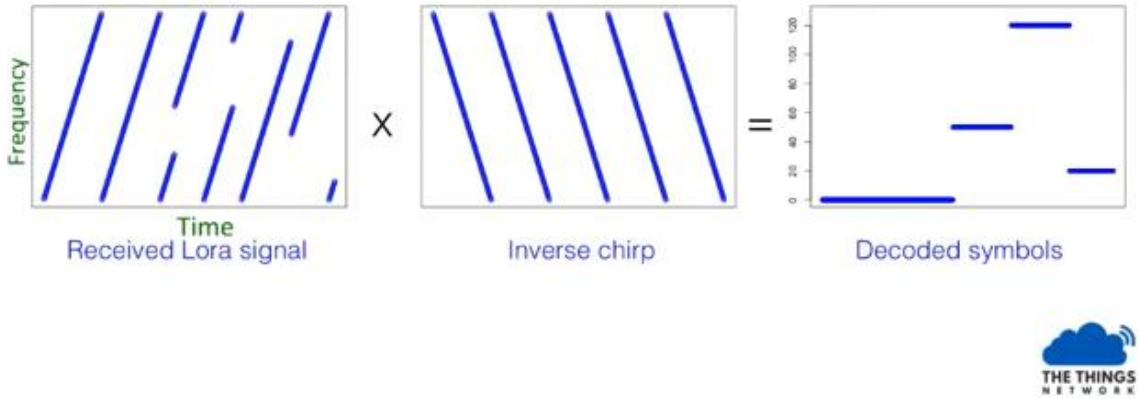


Figura 2.3. Esquema de una red de comunicaciones LPWAN

En el dispositivo receptor se genera el chirp inverso y se multiplica por la señal recibida. De esta forma se obtiene como resultado el símbolo decodificado. Los desplazamientos de fase en la señal recibida son la forma de denotar la transmisión de un símbolo distinto.

En cuanto al tiempo empleado para transmitir los datos, es conveniente introducir el concepto de Spreading Factor (SF), que limita cuantos datos (chirps) se pueden codificar por unidad de tiempo. En el dispositivo SX1276 el SF está limitado entre los valores 7 y 12.

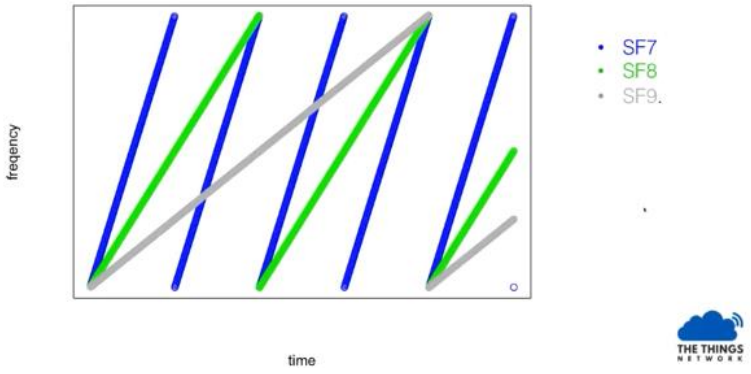


Figura 2.4. Esquema de una red de comunicaciones LPWAN

A la vista de la imagen anterior, cuando se aumenta de SF7 a SF8 se obtienen la mitad de datos por unidad de tiempo y de igual manera para un SF9.

2.4 ELEMENTOS HARDWARE: PLACA DE DESARROLLO NUCLEO STM32 Y TRANSCEIVER LORA

El proyecto se ha desarrollado en una tarjeta "Nucleo STM32F401RE", que cuenta con un microprocesador ARM Cortex-M4 (32 bits), con una memoria flash de 512KB y una RAM DE 96KB. Su frecuencia de operación máxima es de 84MHz [14].

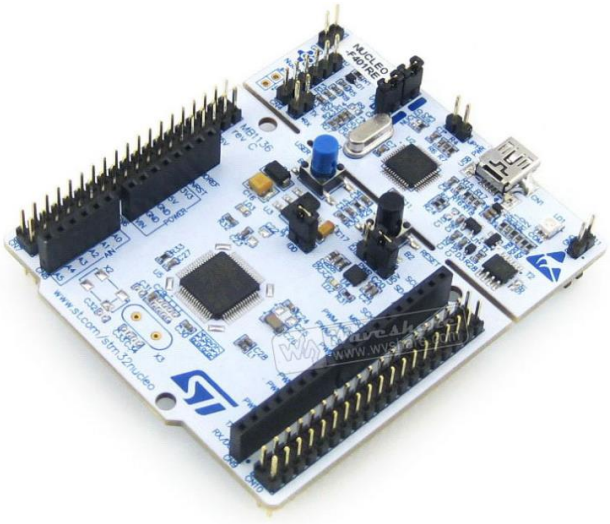


Figura 3. Nucleo board STM32F401RE

Además posee un FPU (Floating Point Unit) dedicada a las operaciones de números en punto flotante, de tal manera que las sumas y multiplicaciones necesarias para realizar el procesado de imagen se realizan de la forma más rápida posible. La FPU ha mejorado las prestaciones de varias funciones de la aplicación, como la transformación del espacio de color y la DCT. La placa también están incluye interfaces (I2C, UART, SPI, USB) para permitir la conexión con periféricos de entrada y salida, así como un convertidor ADC de 12 bits.

Para alimentar el sistema se utiliza la entrada mini-USB a 5V, aunque también se ha evaluado el sistema con una pila de 9V conectada a la entrada CN7 (pin 24) con la tierra al conectada a CN7 (pin 22), teniendo el jumper JP5 colocado en la posición que se indica en la figura adjunta.

Table 7. Power-related jumper

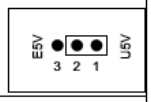
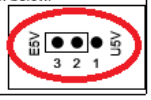
Jumper	Description
JP5	U5V (ST-LINK VBUS) is used as power source when JP5 is set as shown below (Default setting) 
	VIN or E5V is used as power source when JP5 is set as shown below. 

Figura 4. Posición del jumper JP5 para alimentación a través de una pila de 9V.

Si bien es cierto que el procesador puede parecer un tanto insuficiente para hacer una compresión de imágenes en tiempo real, es necesario aclarar que el propósito del proyecto es utilizarlo en aplicaciones en las cuales no sea necesaria una transmisión continua y con alto número de imágenes por segundo, sino que está orientado a la comunicación ocasional de imágenes. Por esta razón, no hay un requisito temporal estricto que obligue a comprimir en tiempo real, lo que estaría también fuera de las posibilidades de comunicación del transceptor de radio, como veremos a continuación. Como ejemplo de aplicación, este proyecto podría formar parte de un sistema en el cual se realizarían recordatorios visuales al usuario mediante una aplicación móvil que incluye una imagen que haya podido captar una cámara, como puede ser un producto en mal estado dentro de la nevera o el estado de la habitación cuando suena la alarma.

Para la difusión de la imagen se utilizarán dos transceptores Lora SX1276 funcionando a una frecuencia de 868 MHz.

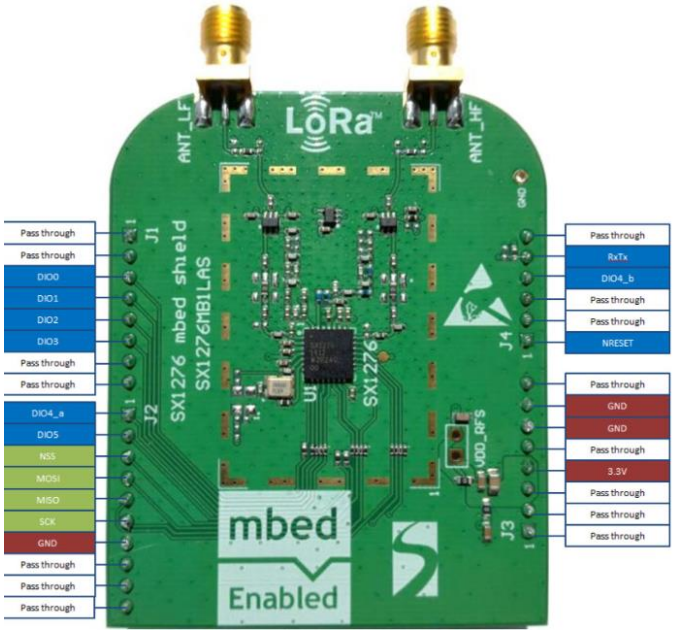


Figura 5. Transceiver SX1276 Lora.

Sus principales características son [6]:

- Atenuación máxima recuperable de -168 dB.
- Potencia máxima de 20 dBm.
- Velocidad de transmisión máxima de 300 Kbps.
- Sensibilidad máxima de -146,5 dBm.
- Corriente en Sleep Mode de 0,2µA.

Se verificará el alcance real del transceptor, que el fabricante fija en 10 Km, pero del que no da más detalles sobre el entorno en el cual se han hecho esas medidas. Es de esperar que el alcance real del dispositivo se vea mermado en un entorno urbano, más conflictivo probablemente que en el del desarrollo del fabricante.

3 DESCRIPCIÓN DEL DISEÑO

El sistema está formado por dos partes: compresor JPEG y la parte de comunicación de datos entre los módulos o entre el PC y el módulo. Todo el desarrollo ha sido llevado a cabo en lenguaje C estándar. El desarrollo incluye tano el código embebido para ser ejecutado en el microprocesador NUCLEO STM32F401-RE como el que se ejecuta desde el PC y que va transmitiendo los valores de los pixeles. En el PC se ha utilizado el entorno de desarrollo Microsoft Visual Studio 2015, mientras que para la plataforma embebida se ha empleado el compilador online de Mbed (www.developer.mbed.org/developers/).

3.1 COMUNICACIÓN PC-EMBEBIDO

En el apartado de especificaciones se indicó que al tener que transmitir una imagen de resolución 640x480 pixeles (profundidad de 3 bytes por pixel: 640*480*3=900kB) no era posible guardar la imagen al completo en el sistema embebido. Es por esto que desde el PC se transmite una porción de imagen correspondiente de resolución de 320x8 (3 bytes por pixel) y esta es la que se va procesando, es decir, se le aplica la compresión JPEG y por último se transmite. Cuando se ha terminado de transmitir esta porción de imagen, el PC vuelve a transmitir por el puerto serie otra porción de imagen. La razón por la que se transmite esa cantidad de información (8*320*3=7680 bytes) es porque ese es el tamaño máximo del buffer que se puede crear en la aplicación. Como es lógico, este proceso debe ser repetido un cierto número de veces hasta completar la transmisión de una imagen. Cada vez se procesan 8 filas y 320 columnas, lo que corresponde (en número de pixeles) a 4 filas completas de una imagen de 640x480 pixeles. Si dividimos 480/4 se obtiene que el proceso debe repetirse 120 veces.

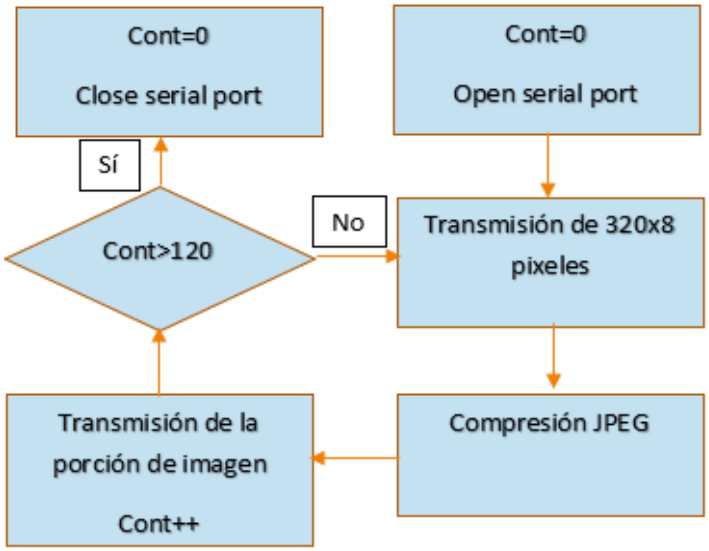


Figura 25. Diagrama de flujo de funcionamiento del sistema de transmisión de imagen.

El módulo comunicación PC-Embebido se encarga de abrir el puerto serie y configurarlo con una de la tasa de 128000 bps y un formato de bytes determinado. En este caso, se van a mandar tramas formadas de 8 bits de información, 1 bit de stop y sin bit de paridad.

Los caracteres leídos correspondiente a los valores de los pixeles se van guardando en una matriz tridimensional (para las componentes luma y croma) de tipo “short int”, de manera que ocupen el menor espacio posible en memoria. No es posible trabajo con tipos “char”, ya que el procesado posterior va a desbordar el tamaño de 8 bits. Una vez el PC ha escrito los 7680 bytes, se leen esos caracteres uno a uno y se van guardando en la posición de la matriz que corresponde. Además, previamente se debe realizar una comprobación de si el canal está disponible para ser leído con el fin de evitar la lectura de caracteres indeseables.

```
short int m[HEIGHT][WIDTH][3];

for(i=0;i<8;i++){
   for(j=0;j<320;j++){

            while(!pc.readable());
                m[i][j][0]=(char)pc.getc();

            while(!pc.readable());
                m[i][j][1]=(char)pc.getc();
            while(!pc.readable());
                m[i][j][2]=(char)pc.getc();

    }
}
```

Figura 26. Introducción de los caracteres transmitidos por el puerto serie a la matriz tridimensional ‘m’.

Una vez cumplido este paso el embebido pasa a realizar la compresión y a transmitir la información por el canal, pero debe haber alguna sincronización para que el PC sepa cuándo tiene que volver a transmitir la siguiente porción de imagen, o de lo contrario podrá darse el caso de que se desborde el buffer del embebido o éste no tenga datos para procesar y transmitir.

Para que esto no suceda, cuando la imagen ya ha sido transmitida el nucleo STM32F401-RE manda un carácter que da paso al PC para que envíe la siguiente parte de la imagen. El acceso a los pixeles por parte del ordenador debe hacerse de tal manera que la imagen sea transmitida en orden para que pueda ser recuperada por el decodificador. Por ello se comienza a transmitir la imagen desde su borde superior izquierdo; las 8 primeras filas y hasta la mitad de la resolución horizontal (320x8) para seguir en la siguiente tanda con las mismas 8 filas y la mitad de la resolución horizontal que no ha sido mandada.

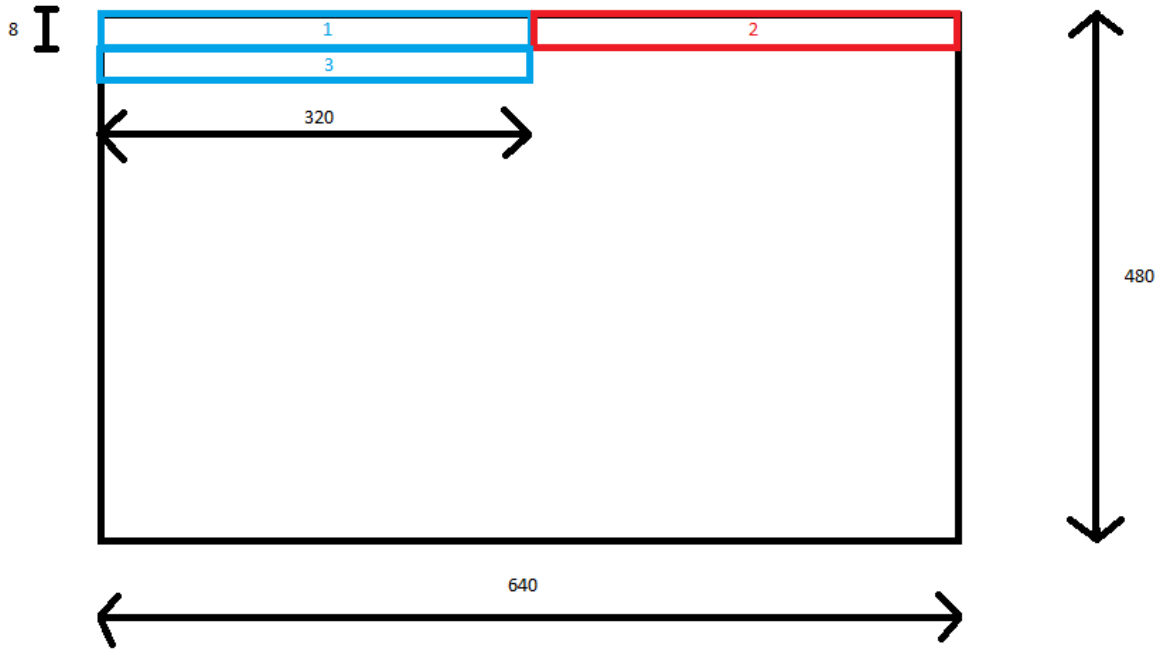


Figura 27. Esquema del orden en el que se va transfiriendo la imagen.

Se utilizan dos índices para acceder a los pixeles adecuados. Uno es 'py' que indica la fila, y el acceso a la columna depende del valor de "desp". Este último oscila entre los valores de 0 y 320 dependiendo de si se debe acceder (observar la figura 27), a la porción de imagen '1' o '2'.

```
for(veces=0;veces<120;veces++){
    if(veces impar)desp=WIDTH/2;
    else desp=0;
    for (i=0;i<8;i++){
        for (j=0;j<desp=WIDTH/2;j++){
            pixel = (Uint8 *)(((imagen)->pixels)) + ((j + desp)*bytes_por_pixel) + (py)*bytes_por_linea;
            //llenar buffer
        }
        py++;
    }

    if(veces par ) py-=8;

    //lectura de carácter de sincronismo.
}
```

Figura 28. Extracto de código para obtener el puntero al pixel indicado según las especificaciones.

3.2 COMPRESOR JPEG

Como se ha explicado anteriormente, el compresor necesita de una serie de procesos que han sido implementados con una función cada uno, y que van a pasar a ser descritas a continuación. Estas funciones son:

1. Conversor RGB a YCbCr.
2. Sub-muestreo de croma.
3. Transformada del coseno.
4. Cuantización.
5. Ordenación en zig-zag.
6. Coding.

3.2.1 Conversión RGB a YCbCr

La primera operación a realizar es la transformación del espacio de color de RGB a YCbCr, para lo cual se ha utilizado las expresiones utilizadas para aplicaciones de televisión estándar según la ITU-R BT.601 [2].

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Ranges:
R/G/B [0 ... 255]
Y [16 ... 235]
Cb/Cr [16 ... 240]

Figura 23. Ecuaciones en forma matricial para la conversión de color de RGB a YCbCR [3].

Al estar los valores de los pixeles en el rango [0,255] se pueden implementar en una variable de tamaño igual a 1 byte. Esto es de ayuda en la transmisión de los pixeles desde el PC a la plataforma embebida mediante el puerto serie a través de una variable de tipo carácter. La función recibe como único argumento un puntero a la imagen que debe procesar (que se describe como una superficie, SDL_Surface) y va recorriendo dicha imagen pixel a pixel, calculando su equivalente en las componentes de luma y croma. Se retorna el puntero a la misma imagen pero con la conversión realizada.

```
for filas{
    for columnas{
        Cálculo de valores de luma y croma.
        Comprobación de que los valores están en los límites según la norma
    }
}
```

Figura 24 Pseudocódigo para la conversión de color de RGB a YCbCR.

Para comprobar que el efecto de cambiar el espacio de color es el correcto se guarda la imagen para comprobar cuál es el resultado y se comprueba mediante otro programa que todos los valores están comprendidos en los límites que marcan la norma.

A primera vista se puede certificar que la mayoría de la información de la imagen no se ha perdido ya que esta sigue siendo distinguible pero se han cambiado las componentes de color. La imagen ha sido guardada con la componente, la croma azul (Cb) en el pixel verde y la croma roja (Cr) en el pixel azul.

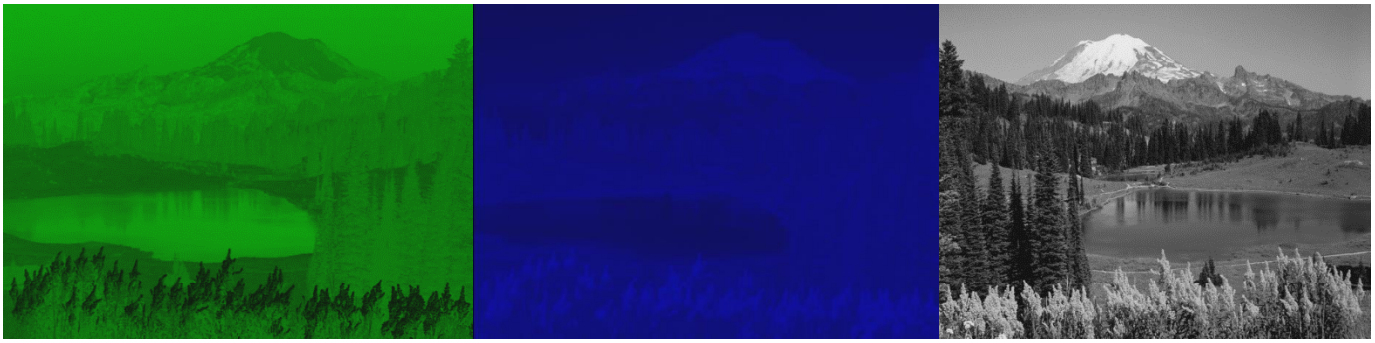


Figura 26. Imágenes mostrando las componentes YCbCr por separado guardados sobre una imagen RGB.

La imagen con un nivel mayor de detalle es la que contiene la información de luma (imagen roja). Esta observación justifica que se utilicen todos los componentes de luma y que el sub-muestreo de croma pueda omitir parte de la información de color ya que no repercutirá (excesivamente) en una pérdida de calidad de imagen. Especialmente significativa es la poca información que se guarda en la componente de croma roja (imagen azul), en la cual apenas se aprecia el paisaje origina

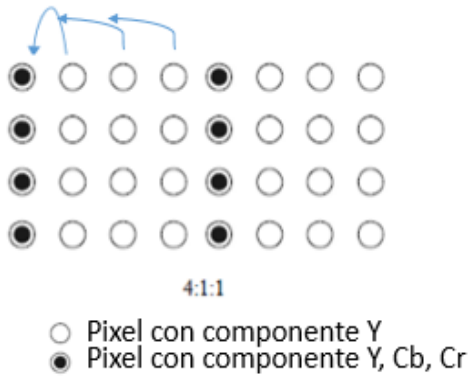
3.2.2 Sub-muestreo de croma

Se va a comparar en este apartado la pérdida de calidad con cada uno de los sub-muestreos más comunes, los cuales han sido introducidos en el diseño. Se van a estudiar 4 casos de sub-muestreo:

1. 4:1:1
2. 4:2:2
3. 4:4:0
4. 4:2:2

Las porciones de código que llevan a cabo estas conversiones tienen en común que van recorriendo la imagen, ya no pixel por pixel como en el apartado anterior, sino que dependerá de la resolución que ese tipo de sub-muestreo provea al plano horizontal o vertical. Por ejemplo para el sub-muestreo 4:4:0, la resolución horizontal es completa pero la vertical es de 1/2. Esto significa que las filas de la imagen se deben recorrer por pares (aumentando en 2 la variable que indique la fila de la imagen) y las columnas de una en una. Este sub-muestreo (al ser opcional) no está implementado como una función a parte sino como un añadido a la función de conversión de RGB a YCbCr.

1. El sub-muestreo 4:1:1 provee un resolución horizontal de 1/4 mientras que la vertical es completa. Por lo tanto se va a hacer un recorrido línea a línea por la imagen mientras que las columnas se saltan de cuatro en cuatro pixeles.

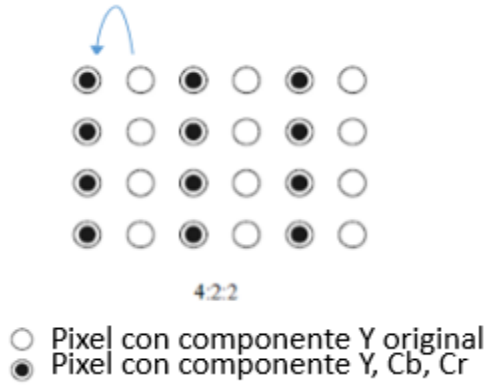



```
    //sub sampling 4:1:1 extract
    for(i=0;i<imagen->h;i++){
        for(j=0;j<imagen->w;j+=4){
/*
 * 3 Bytes per pixel
 * Y = *(pixel+2)
 * Cb=*(pixel+1)
 * Cr=*(pixel);
 */
            pixel=(Uint8 *)((imagen->pixels))+(j*bpp)+i*linea;
            //first pixel of the row is not modified

            //second pixel is equal to the first one (luminance)
            *(pixel+3)=*(pixel);
            *(pixel+4)=*(pixel+1);

            //third pixel
            *(pixel+6)=*(pixel);
            *(pixel+7)=*(pixel+1);

            //fourth pixel
            *(pixel+9)=*(pixel);
            *(pixel+10)=*(pixel+1);
        }
    }
```

Figura 27. Esquema de información contenida en cada uno de los pixeles bajo sub-muestreo 4:1:1y código para llevarlo a cabo..

El primero de los pixeles se deja intacto y para los demás, las componentes de croma se igualan a las de ese primer pixel. No debe olvidarse que la luminancia no puede ser modificada en ningún caso ya que aquí reside la mayor fuente de información, por lo que un sub-muestreo se percibiría rápidamente por la pérdida de calidad de imagen.

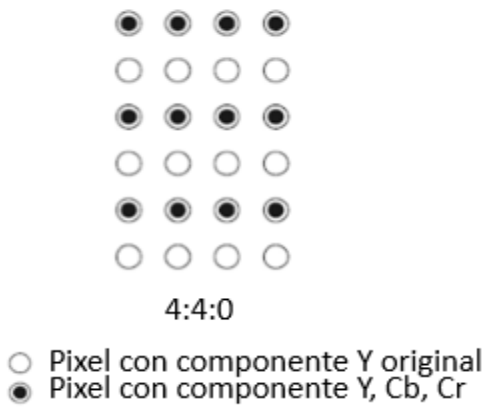
2. El sub-muestreo 4:2:2 también provee una resolución vertical completa, mientras que la horizontal es de 1/2, es por esto que se debiera percibir una mejora en la calidad respecto al caso anterior. Por ello, en el código se va a percibir cómo de igual manera los incrementos verticales son idénticos pero horizontalmente el desplazamiento se realiza por pares de pixeles.




```
//sub sampling 4:2:2 extract
for(i=0;i<imagen->h;i++){
    for(j=0;j<imagen->w;j+=2){
        pixel=(Uint8 *)((imagen->pixels))+(j*bpp)+i*linea;

        //second pixel of row is equal to the first one
        *(pixel+3)=*(pixel);
        *(pixel+4)=*(pixel+1);

    }
}
```

Figura 29. Esquema con la información original contenida en cada pixel y código necesario para llevar a cabo un sub-muestreo 4:2:2.

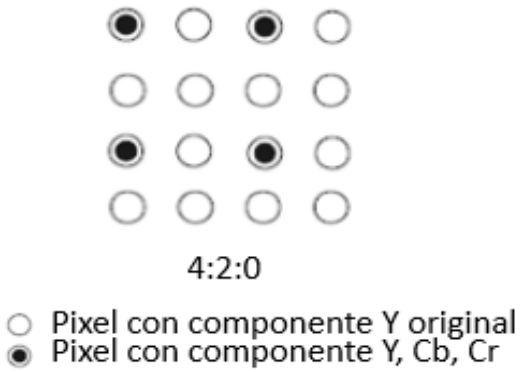
3. Para el caso del 4:4:0 hay un ligero cambio respecto a los anteriores sub-muestreos y es que se introduce un pérdida de resolución vertical y no horizontal, es por esto que la imagen ha de ser recorrida pensando en asignar a ciertos pixeles los valores de croma correspondiente al de la fila inmediatamente superior.




```
//sub sampling 4:4:0 extract
for(i=0;i<imagen->h;i+=2){//processing lines in groups of two
    for(j=0;j<imagen->w;j++){
        pixel=(Uint8 *)((imagen->pixels))+(j*bpp)+i*linea;
        Uint8 temp1,temp2;
        temp1=*(pixel);
        temp2=*(pixel+1);//save values

        pixel=(Uint8 *)((imagen->pixels))+(j*bpp)+(i+1)*linea;

        *(pixel)=temp1;//assign values of previous line
        *(pixel+1)=temp2;

    }
}
```

Figura 30. Esquema con la información original contenida en cada pixel y código necesario para llevar a cabo un sub-muestreo 4:4:0.

4. El último tipo de sub-muestreo es el 4:2:0 que reduce de igual manera la resolución del plano vertical y el horizontal a la mitad, por lo que se hará una comparación para contrastar la diferencia de calidad respecto al ejemplo con 4:2:2.




```
//sub sampling 4:2:0 extract
for(i=0;i<imagen->h;i+=2){
    for(j=0;j<imagen->w;j+=2){
        pixel=(Uint8 *)((imagen->pixels))+(j*bpp)+i*linea;
        Uint8 temp1,temp2,temp3,temp4;
        temp1=*pixel;
        temp2=*(pixel+1);

        *(pixel+3)=*(pixel);
        *(pixel+4)=*(pixel+1);

        pixel=(Uint8 *)((imagen->pixels))+(j*bpp)+(i+1)*linea;

        *(pixel+3)=temp1;
        *(pixel+4)=temp2;

    }
}
```

Figura 31. Esquema con la información original contenida en cada pixel y código necesario para llevar a cabo un sub-muestreo 4:2:0.

Para ilustrar la diferencia de calidad con cada uno de los sub-muestreos se muestran las imágenes recuperadas después de haberlas convertido al formato YCbCr y aplicarlas el comentado muestreo. Es previsible que las imágenes con menor calidad van a ser las del tipo 4:1:1 o 4:2:0, ya que introducen una pérdida de resolución de 1/4 en el plano horizontal y de 1/2 en ambos planos respectivamente.

En el siguiente escalón de pérdida de detalle debería aparecer el 4:2:2 y 4:4:0, que proveen una resolución completa en uno de los dos planos y de la mitad en el otro.

La imagen a procesar para la prueba debe tener un alto nivel de diferentes texturas y formas en las que incida la luz sobre el objetivo, además de un primer plano donde se halle el principal de información y un fondo con algún tipo de detalles.

3.2.3 Transformada del coseno

La DCT se va a aplicar sobre grupos de pixeles de 8x8 llamados macrobloques. La función de entrada va a tener como argumentos:

- Matriz tridimensional de tamaño 8x8x3 (para las tres componentes) que tiene introducidos los valores de los pixeles del macrobloque en cuestión-
- Matriz tridimensional de tamaño HEIGHTxWIDTHx3 en la que se almacenan los valores de la transformada del coseno de todos los macrobloques de la imagen.
- Dos variables que indexan en qué fila y columna de la matriz de mayor tamaño hay que almacenar los valores de la DCT del macrobloque a procesar.

Para verificar que los resultados de la transformada son los esperados, se hace uso de la librería de software libre Scilab, de manera que se comparan mediante un fichero la salida correspondiente al cálculo con el compresor JPEG y el obtenido mediante la función de referencia ("Golden Model") "dct" de la libraría Scilab. Este software de referencia proporciona soluciones con mayor precisión que el desarrollado y, por tanto distintas. Por ello, se ha tolerado un margen de error de un 10%. El algoritmo en pseudocódigo para llevarlo a cabo se muestra a continuación.

```
read input
matrix2=dct(input,-1)

read matrix (result from C algortihm)
for i=0 to HEIGHT
    for j=0 to WIDTH
        if(10%matrix2(i,j) + matrix2(i,j) > matrix(i,j) && matrix2(i,j) - 10%matrix2(i,j) < matrix(i,j)) good!!
        else error!!!
    end
end
```

Figura 36. Pseudocódigo para cheque del resultado de DCT.

3.2.4 Cuantización

En base a las tablas de cuantización presentadas en el apartado de anterior, se va realizando la cuantificación por división de los valores obtenidos de la DCT. Es de especial interés el contraste en los ratios de compresión obtenido en base a la modificación de este parámetro. Se acentúa sobremanera el hecho de que la luma es el componente que contiene más información y cómo una reducción de valores significativos (distintos de cero) tiene un efecto muy positivo sobre el factor de compresión. Se va a probar el compresor con valores en las tablas de menor tamaño, que haga que un menor número de valores sea cero y el procedimiento contrario.

El único parámetro que se le pasa a la función encargada de realizar este paso es la matriz con la información de la DCT, siendo las tablas locales a la función.

3.2.5 Ordenación en zig-zag.

Hasta ahora los datos se almacenan en una matriz pero la ordenación en zig-zag requiere otro espacio en memoria donde almacenar el resultado. Para ello se hace una reserva de espacio dinámica con la instrucción “malloc“. El algoritmo de la ordenación en Zig-Zag utiliza dos punteros para indicar la fila y columna de la imagen, que deben ser variados en función de la ordenación. En definitiva hay dos tipos de movimientos: diagonal ascendente (hacia la derecha) y diagonal descendente (hacia la izquierda). Por ello se introduce una variable para saber qué tipo de movimiento se está implementado en un momento concreto.

3.2.6 Coding.

Esta función, que tiene como único argumento de entrada el puntero al vector donde están todos los elementos después de la ordenación, será llamada dos veces. Hay que recordar que este tipo de codificación aprovecha la redundancia en la fuente y envía cuántas veces aparece de manera consecutiva un símbolo y a continuación el símbolo. Al no saberse en tiempo de compilación cuál va a ser la longitud del mensaje (ya que esto depende de la imagen) se llama a la función una primera vez para que haciendo la codificación usual se pueda averiguar el espacio que va a ocupar cada una de las componentes. De esta manera se procede a realizar la reserva de memoria para exactamente los bytes necesarios. Por longitud se entiende la cantidad de valores necesarios para enviar el mensaje.

97	8	0	0
16	0	0	0
0	0	0	0
0	0	0	0

1	97	1	8	13	0

longitud=6

Figura 37. Resultado de aplicar la ordenación en zig-zag y el run length coding a un macrobloque 4x4.

Siguiendo el ejemplo, la primera llamada sería para averiguar esa longitud de seis y hacer una reserva de 12 bytes (6 short int), en la segunda se comenzarían a guardar los valores correspondientes en los espacios de memoria reservados. Seguidamente se libera el espacio reservado previamente como contenedor del resultado del zig-zag.

A continuación se procede a llamar a la función “lora”, que es la encargada de establecer el enlace con el receptor para mandar la imagen comprimida.

Figura 38. Imagen original y después de comprimirla con el compresor JPEG.

También se ha realizado una caracterización del compresor en función de las tablas de cuantificación, viendo cómo se modifica el ratio de compresión en función de la tabla de cuantización y cómo varía la calidad de la imagen.

Bodegón *Lena* *Habitación*

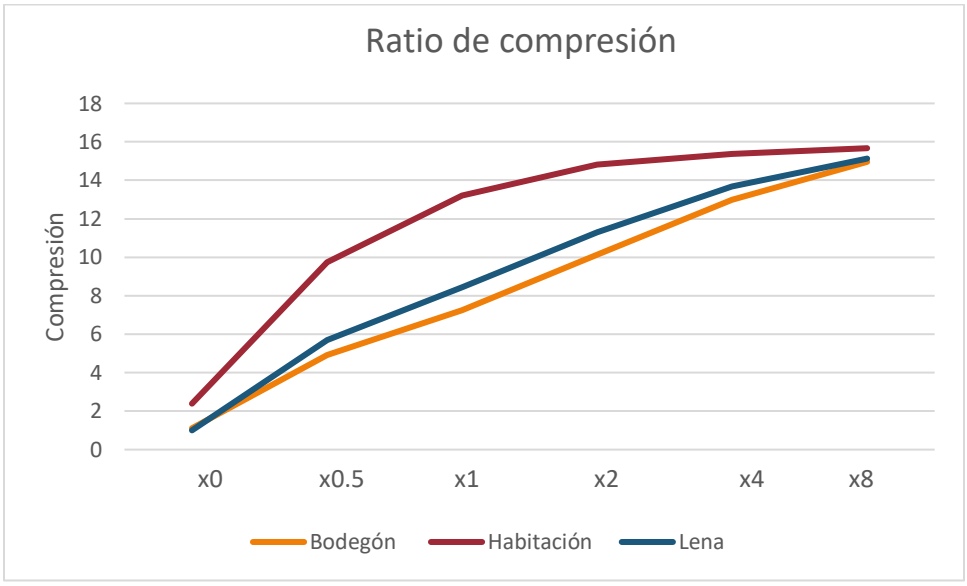


Figura 39. Resultado de aplicar la ordenación en zig-zag y el run length coding a un macrobloque 4x4.

En la figura se muestra cómo el compresor ha actuado sobre 3 imágenes diferentes que se pueden agrupar en 2 grupos:

- Imágenes con alto nivel de texturas y contraste (Lena y Bodegón).
- Imagen con un gran nivel de redundancia (pared).

En el eje vertical se representa el ratio de compresión y en el eje de abscisas se representa el factor por el que ha sido multiplicada la tabla de cuantización que se utilizó originalmente (figura 17, pág. 16).

Como es lógico, la imagen más propicia (habitación), por su mayor uniformidad, produce unos niveles de compresión considerablemente más altos que las otras imágenes, pero se percibe cómo a medida que se aumenta el factor de la tabla de cuantización, la compresión también alcanza un mayor nivel pero llega un momento en el que “satura”. El paso de un factor

2 a un 4 muestra cómo el ratio apenas se incrementa, pero la calidad de la imagen se resiente enormemente. El límite de compresión de la imagen roza un valor de 16.

En cambio, las imágenes con alto nivel de detalle muestran un comportamiento lineal hasta llegar a un factor x4. En la última transición de cambio en el factor de cuantización de x4 a x8 se percibe cómo la compresión pierde ligeramente la linealidad, no obstante no llega al grado de saturación que con la imagen anterior.

3.3 COMUNICACIÓN DE RADIO ENTRE TRANSCEIVERS.

Habiendo realizado el traspaso de la imagen del PC al STM32F401-RE y hecho el procesado de esta, sólo queda enviar la imagen por radio. Se implementa esta tarea mediante una función que tiene como parámetros de entrada los punteros a los vectores en los que se almacena cada una de las componentes (YCbCr) codificadas, además de otro escalar que indica la longitud de este vector. Cabe aclarar que el código es que se carga tanto en el dispositivo que actúa de transmisor como el que actúa de receptor es el mismo salvo una variable booleana que dependiendo de su valor hará que se ejecute una parte u otra en base a sentencias de control. La razón de por la que se emplea esta forma de diseño es que se va a hacer una implementación con un módulo trasmisor/receptor (transceiver), por lo que dentro del programa debe haber alguna forma de intercambiar el rol de transmisor y receptor.

La función se puede dividir principalmente en dos bloques, que son:

- Configuración del transceiver para trabajar a una determinada frecuencia, establecer la potencia de salida, el ancho de banda, Spreading Factor, y formato de los paquetes a enviar.
- Máquina de estados software encargada de implementar un protocolo de comunicaciones que garantice la transmisión de la imagen.

3.3.1 Configuración del transceiver.

La frecuencia de trabajo es de 868 MHz, ya que para ese valor está optimizada la antena utilizada. Si se quiere trabajar a distancias largas, es necesario fijar la potencia de salida al máximo (20dBm).

La sensibilidad del receptor viene dada por la siguiente expresión:

$$S = -174 + 10\ \log_{10} BW + NF + SNR \qquad \text{Eqn. 1}$$

- El primer término corresponde al ruido térmico en 1 Hz, el cual puede ser modificado únicamente cambiando la temperatura.
- BW es el ancho de banda.
- NF es la figura de ruido del receptor.
- SNR es la relación señal a ruido que se desea obtener.

Los dos únicos términos sobre los que tiene poder el diseñador son el ancho de banda y la relación señal a ruido.

La principal premisa en la que se basa la codificación Lora es que cada bit se codifica como un conjunto de múltiples pulsos (chips). La relación entre el bit rate y el chip rate tiene la siguiente expresión:

$$R_b = SF * \frac{\left[\frac{4}{4+CR}\right]}{\left[\frac{2^{SF}}{BW}\right]} \qquad Eqn.\ 2$$

Siendo SF el Spreading Factor (valor entre 7 y 12) , BW el ancho de banda (en Hz), CR el code rate (valor entre 1 y 4) y la velocidad de transmisión oscila entre 0.3kbps y 22 kbps. Además, también es capaz de modificar el valor del SNR como se muestra a continuación:

Modulation	Typical SNR
LoRa SF12	-20 dB
LoRa SF10	-15 dB
GMSK	9 dB

Figura 40. Tabla con valores típicos de SNR para un cierto SF.

Los valores negativos del SNR indican que es capaz de recuperar una señal que está por debajo del nivel de ruido.

El ancho de banda se puede fijar en tres valores, que son 125KHz, 250 KHz y 500KHz, con el SF con valores comprendidos entre 7 y 12.

También está a disposición del usuario la utilización de distintos "Coding rates" (CR), los cuales suponen una sobrecarga de bits que permite la recuperación de información debido a problemas de interferencia. Las diferentes posibilidades son 4/5, 4/6, 4/7, 4/8, en el que el numerador indica el número de bits de información y el denominador el total de bits codificados, por lo que hay una información redundante igual a la diferencia entre ellos.

En el apartado de "Evaluación" se van a hacer medidas del radioenlace a diferentes distancias para verificar cómo influyen el Spreading Factor y el ancho de banda sobre la sensibilidad. Así mismo, el SF tiene un impacto sobre el tiempo que va a ser necesario para transmitir un paquete ya que este parámetro define con cuántos chips se van a codificar cada bit de información. Cuanto mayor sea esta relación, mayor tiempo se requerirá para completar la transmisión (para un mismo ancho de banda).

El formato de los paquetes consta de un preámbulo, cabecera (opcional) y la información (payload) [11].

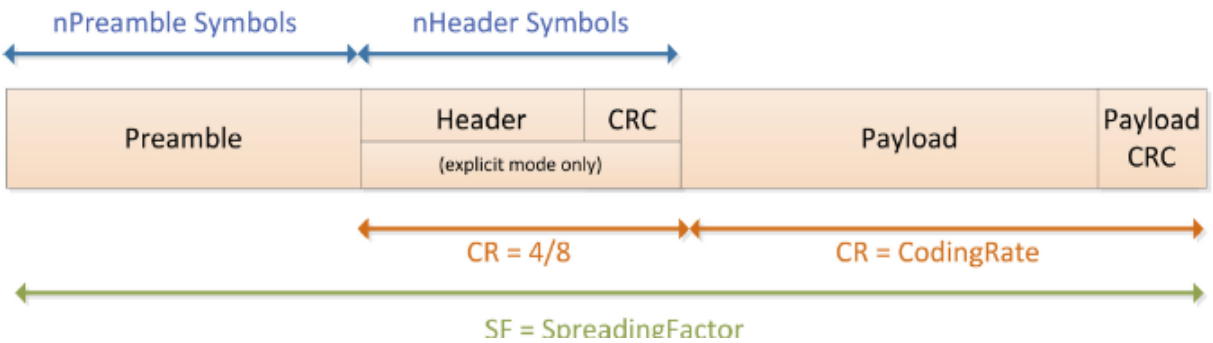


Figura 41. Formato paquetes Lora.

Es útil definir el tiempo que se tarda en transmitir un símbolo, el cual viene definido por el tiempo necesario para transmitir 2^{SF} chips al chip rate utilizado (es igual al ancho de banda).

$$T_{sym} = \frac{2^{SF}}{BW}$$

Figura 42. Expresión del tiempo de símbolo.

El tiempo de paquete será la suma del tiempo del preámbulo más el de la información:

$$T_{packet} = T_{preamble} + T_{payload}$$

$$T_{payload} = payloadSymbNb \ \ Tsym \qquad T_{packet} = T_{preamble} + T_{payload}$$

Siendo cada uno de estos tiempos el tiempo de símbolo por el número de símbolos que englobe. El número de símbolos del payload viene dada por la siguiente expresión:

$$payloadSymbNb = 8 + max\left(ceil\left(\frac{8PL - 4SF + 28 + 16 - 20H}{4(SF - 2DE)}\right)(CR + 4), 0\right)$$

- PL es el número de bytes de la carga.
- SF es el Spreading Factor.
- CR es el Coding Rate.
- H=0 si hay cabecera, si no es 1.
- DE=1 si está activa la optimización para bajas velocidades de transmisión, si no 0.

La siguiente gráfica muestra la cantidad de símbolos contenidos en un paquete en función del Spreading Factor. De las cuatro líneas, las dos inferiores se refieren a PL=32 y las superiores para PL=64, además se muestra la diferencia de utilizar la optimización para bajas tasas de transmisión de bits.

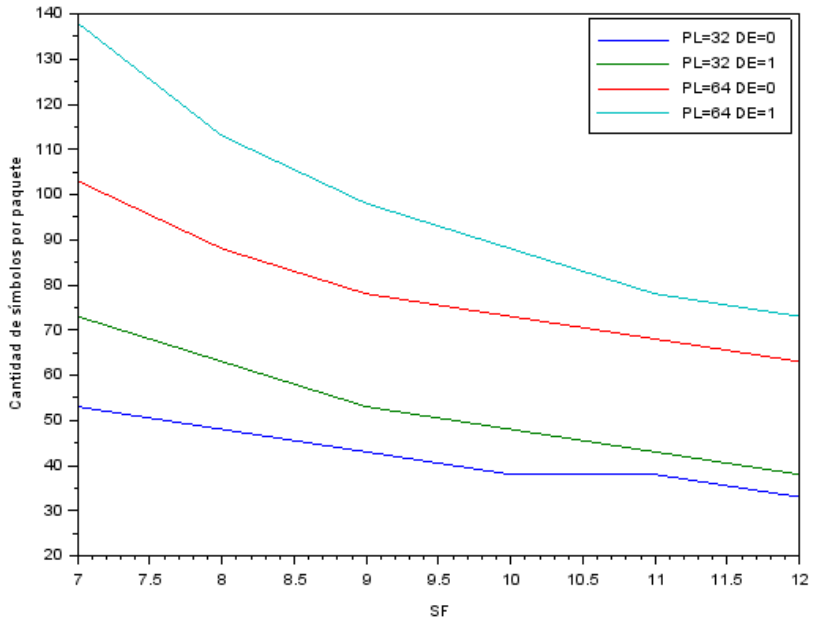


Figura 43. Relación de los símbolos contenidos en un paquete respecto al Spreading Factor.

La optimización Lora mejora considerablemente la cantidad de símbolos contenidos en un paquete, pero la principal diferencia está en los bytes que se tienen de información (PL). Al aumentar este parámetro, la relación entre los símbolos introducidos por paquete y la parte no útil a efectos de información (cabecera y preámbulo) aumenta, mejorando la eficiencia.

Además, también se clarifica cómo el SF tiene un efecto depresor sobre el tiempo que se va a necesitar para enviar un mensaje ya que al aumentar este parámetro, el tiempo por símbolo incrementa exponencialmente y esto se ve reflejado en que los símbolos por paquete se ven reducidos. En la siguiente tabla se muestra los valores (en milisegundos) de la duración del paquete desglosados en el tiempo del preámbulo y el de la información para diferentes valores de SF y BW.

	PL=32 DE=0			PL=32 DE=1		
	$T_{preamble}$	$T_{payload}$	T_{packet}	$T_{preamble}$	$T_{payload}$	T_{packet}
SF=7 BW=500kHz	3,14	13,57	16,70	3,14	18,69	21,82
SF=10 BW=250kHz	50,18	155,65	205,82	50,18	196,61	246,78
SF=12 BW=125kHz	401,41	1081,34	1482,75	401,41	1245,48	1646,59

	PL=64 DE=0			PL=64 DE=1		
	$T_{preamble}$	$T_{payload}$	T_{packet}	$T_{preamble}$	$T_{payload}$	T_{packet}
SF=7 BW=500kHz	3,14	26,37	29,50	3,14	35,33	38,46
SF=10 BW=250kHz	50,18	299,01	349,18	50,18	360,45	410,62
SF=12 BW=125kHz	401,41	2064,38	2465,79	401,41	2392,06	2793,47

Tabla 1. Tiempos necesarios para enviar un paquete para distintas configuraciones de SF y BW.

Los resultados muestran cómo tanto con optimización (DE=1) o sin ella (DE=0), es más conveniente utilizar un tamaño de paquete de 64 bytes, ya que maximiza la cantidad de información enviados por unidad de tiempo.

3.3.2 Máquina de estados software

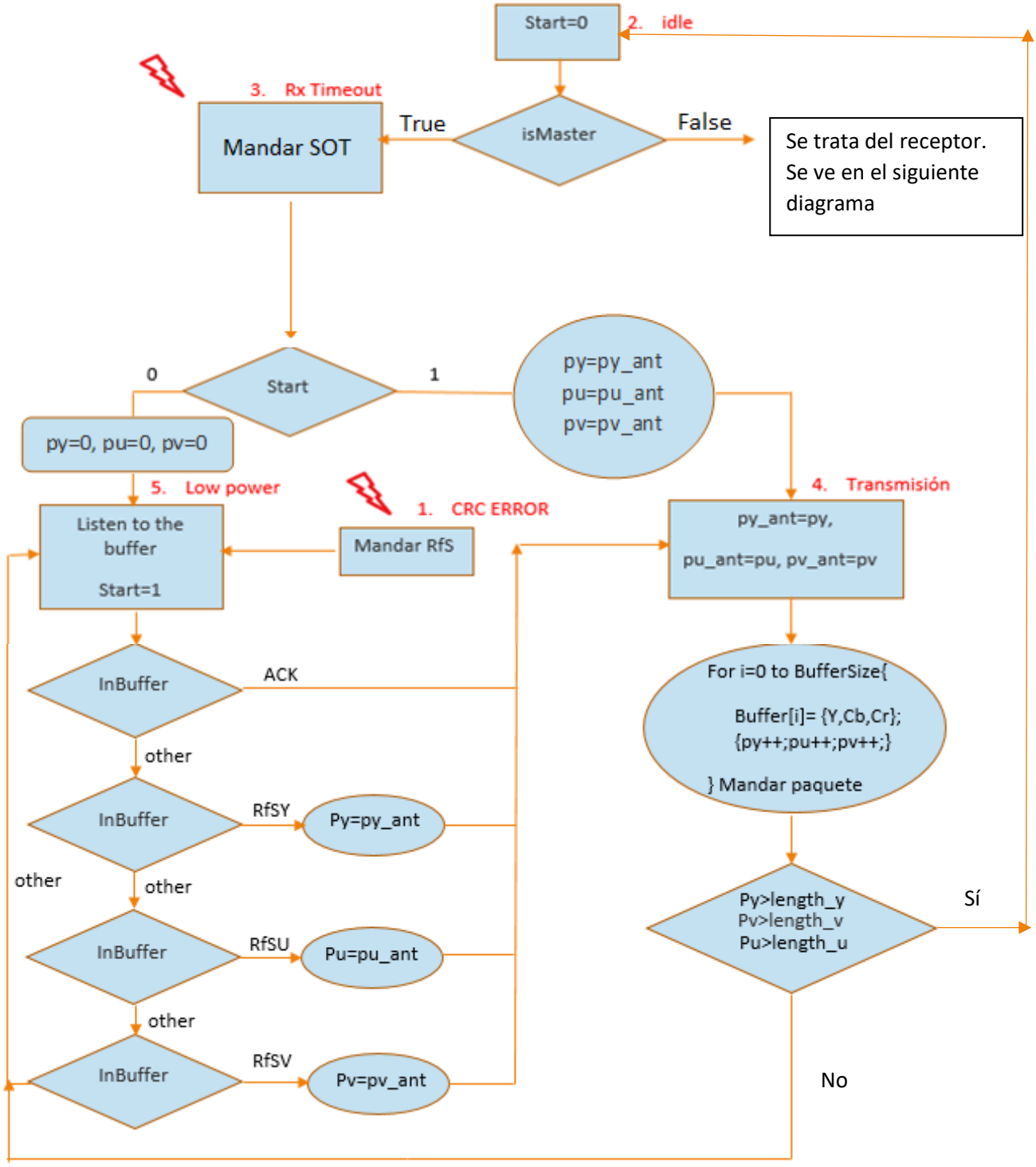
Si bien el concepto de máquina de estados se suele referir a una descripción de hardware en la que se le indica a un circuito cómo debe responder ante un flanco de reloj dependiendo típicamente de unas entradas, en esta ocasión se va a utilizar con un lenguaje de programación software. Por ello, no se va a tener un control tan de bajo nivel pero sí que se van a definir estados y, en función de ciertas entradas, se van a tomar unas decisiones u otras. El modelado mediante máquinas de estados de la funcionalidad software es de uso común en metodologías software basadas en modelos como UML (Unified Modeling Language). Además, la programación basada en máquinas de estados permite una implementación eficiente de código en sistemas embebidos.

Como bien he aclarado, los estados definidos deben ser capaces de responder ante todas las posibles situaciones en las que se encuentre el transceptor:

- Transmisión
- Recepción.
- Recepción de un paquete con errores.
- Modo Sleep (bajo consumo).

Los estados en ambos transceptores son los mismos pero las acciones serán obviamente distintas. El objetivo es definir un protocolo de comunicaciones que asegure la correcta transmisión de la información al destino.

El siguiente diagrama de flujo muestra el funcionamiento simplificado general del código utilizado para implementar en el transmisor y en el receptor:



Para poder entender en profundidad el diagrama es necesario describir las variables que se usan en el mismo:

- isMaster es la variable booleana cuyo valor cambiaba para el transmisor y para el receptor. 'True' para el transmisor y 'false' para el receptor.
- Px, py, pu son los índices a los vectores que contienen las componentes YCbCr. Cuando alcancen un valor igual a la longitud de estos vectores quiere decir que ya se ha enviado toda la información referente a esta componente. Cuando las tres componentes han sido enviadas se ha transmitido la imagen.
- Length_y, length_u, length_v indican la longitud de cada uno de los vectores que contienen las componentes de color codificadas.
- Start indica si se ha enviado el primer paquete o de la transmisión o no. Esto se debe a que para este primer paquete los índices a los vectores de las componentes deben estar a su primer valor (cero) y si se da otro caso de un paquete perdido se deben fijar al valor anterio de la transmisión.
- InBuffer es el buffer de entrada.
- ACK es el Acknowledge que se debe recibir por parte del receptor y que expresa que se ha recibido un paquete correctamente.
- RfSY, RfSU, RfSV son los "Request for send" que enviará el receptor cuando se reciba un paquete con errores. La última letra da cuenta de qué componente se está enviando cuando hay un paquete erróneo.

Lo primero que se evalúa es la variable booleana para saber si se está en el transmisor o receptor (idle). Llegado este punto se examina el primer caso por lo que se espera a que salte el timeout en la recepción (por no recibir ningún paquete) para enviar un paquete de SOT (Start of Transmission). Si es el primer paquete que se envía (start=0), se inicializan las variables, que son los índices a los vectores, a 0 y se pasa a un estado de bajo consumo a esperar que llegue la confirmación. Cuando esto ocurra, se va llenando el buffer de salida con los valores de la componente que corresponda, en primer lugar la luma y después las componentes de croma. Es importante guardar los valores de los índices de acceso a los vectores para poder abordar un reenvío en caso de pérdida o error en un paquete. Para finalizar esta parte de la transmisión se comprueba si los índices py, pu y pv han llegado a los valores de la longitud de cada vector que contiene una componente de la imagen. Si esto ocurre es que se ha terminado y debe retornarse de la función.

Mientras se está en ese estado de bajo consumo esperando la respuesta del receptor, es posible que llegue un "Request for send", en este caso se debe proceder a un reenvío del paquete. Por ello se hace una recarga del puntero al vector con el valor previo a la transmisión.

En las siguientes figuras se muestra visualmente cómo responde el protocolo ante algunas situaciones. En cuanto al arranque, como ya se ha comentado, se hace después de que salte el timeout y se mande el SOT.

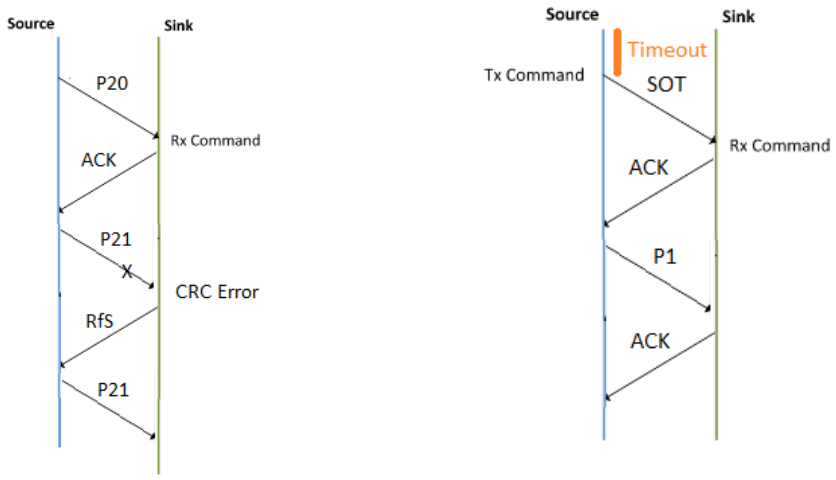


Figura 45. Esquema de inicio de transmisión y de reenviío de paquetes.

En caso de que haya algún paquete perdido o erróneo se procede al reenvío. En cuanto al orden en que se envían la imagen, estos siguen un esquema en el cual se envía previamente la longitud del vector que contiene la componente a mandar para luego mandar la información respectiva a esa componente.

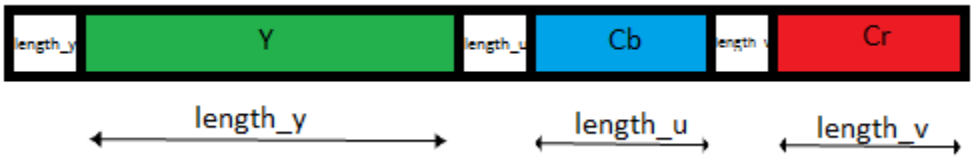


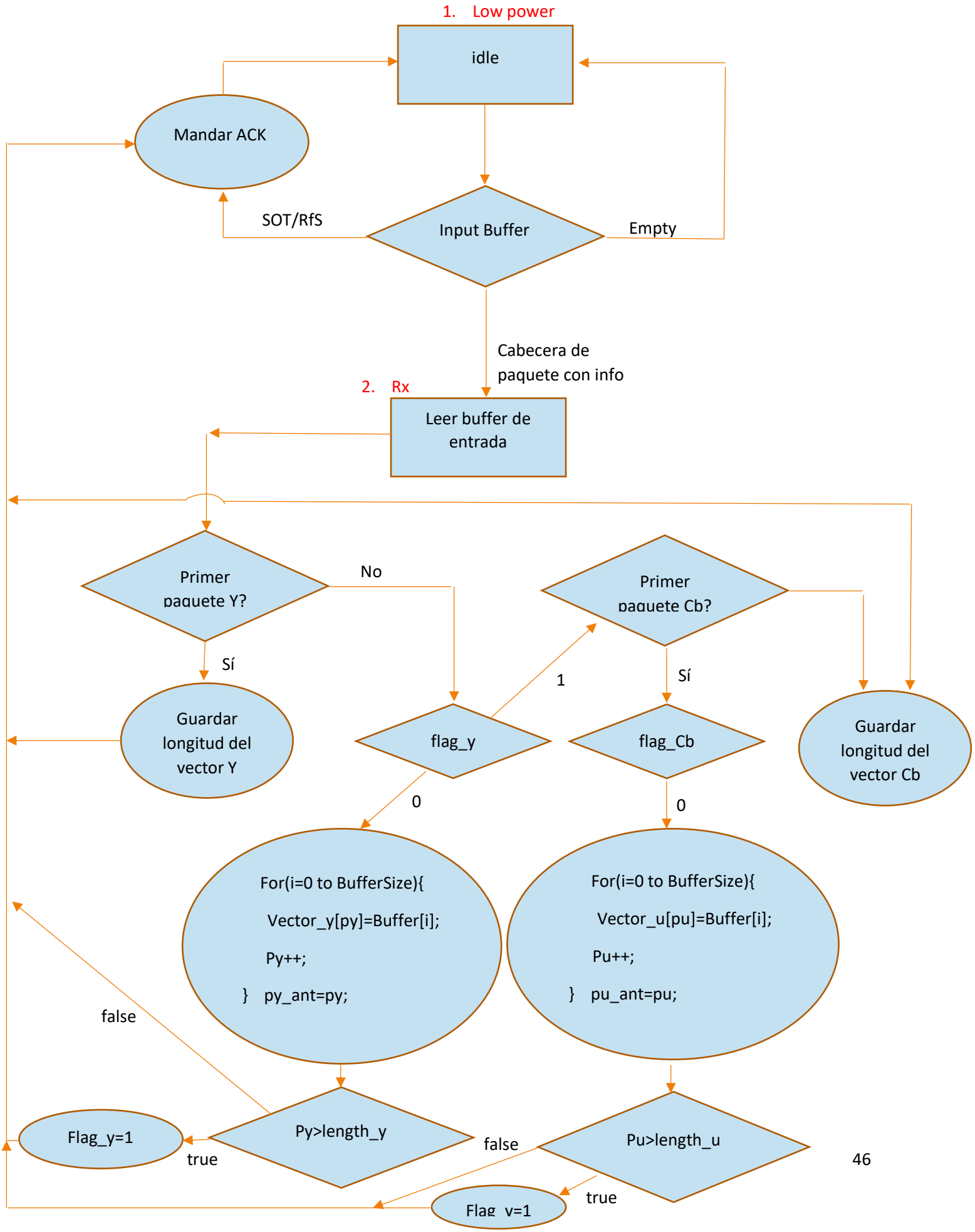
Figura 46. Forma de envío de las tres componentes de una imagen.

En cuanto al llenado del buffer de salida es necesario hacer una adaptación debido a que las componentes YCbCr están guardadas tienen un tamaño de tipo “short int” (2 bytes) y el buffer es de caracteres (1 byte). Por ello, en un byte se carga la parte menos significativa del elemento del vector ‘short int’ y en el siguiente elemento del buffer la parte más significativa. Esto último se consigue con un desplazamiento hacia la derecha de 8 bits.

```
for (i=0;i<BufferSize;i+=2){
    Buffer[i]=vector[py] & 0xff;
    Buffer[i+1]=(vector[py]>>8) & 0xff;
    py++;

}
```

Figura 47. Código necesario para hacer la carga de una variable de tipo short int a dos de tipo char.

El transceiver que va a actuar como receptor de la imagen tiene su propia máquina de estados para gestionar la llegada y almacenamiento de la imagen.


1. Low power
idle
Mandar ACK
SOT/RfS
Input Buffer
Empty
Cabecera de
paquete con info
2. Rx
Leer buffer de
entrada
Primer
paquete Y?
No
Sí
Guardar
longitud del
vector Y
flag_y
1
0
Primer
paquete Cb?
Sí
flag_Cb
0
Guardar
longitud del
vector Cb
For(i=0 to BufferSize){
Vector_y[py]=Buffer[i];
Py++;
} py_ant=py;
For(i=0 to BufferSize){
Vector_u[pu]=Buffer[i];
Pu++;
} pu_ant=pu;
false
Flag_y=1
true
Py>length_y
false
Pu>length_u
true
Flag_v=1

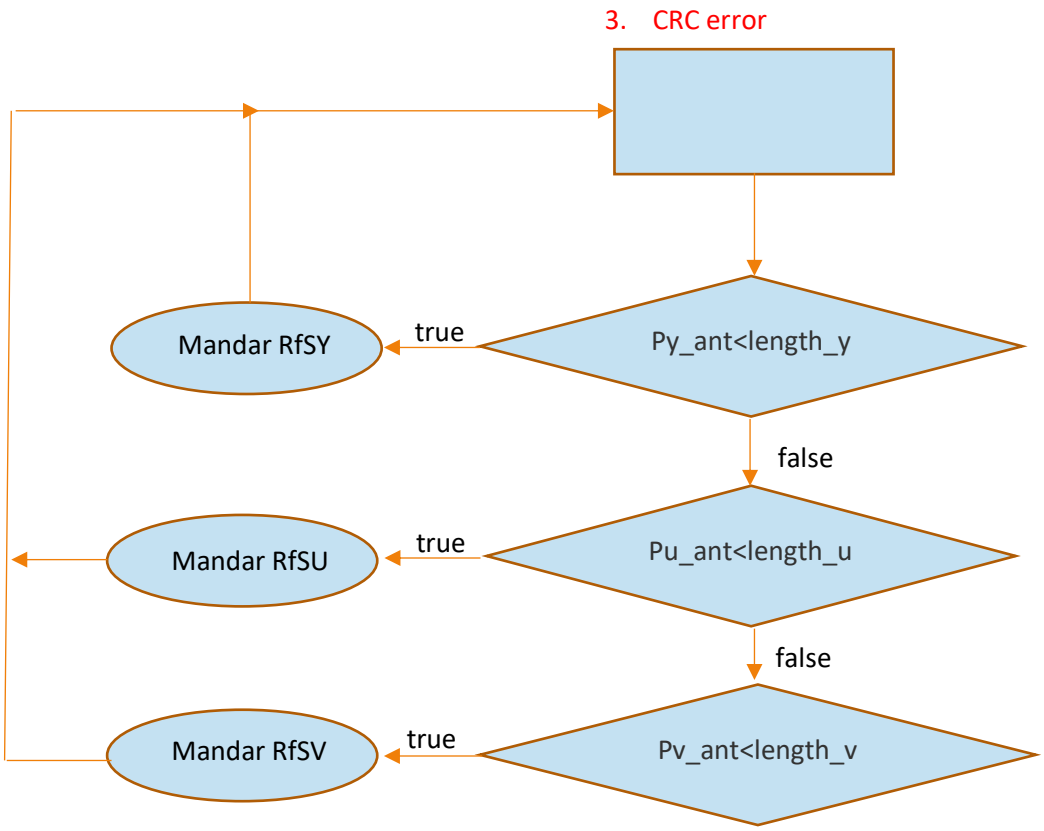


Figura 48. Diagrama de funcionamiento del programa implementado en el transceiver que actúa como receptor.

**El diagrama sólo incluye el proceso de lectura para las componentes Y Cb. Por falta de espacio no se incluya la Cr pero seguiría el mismo procedimientos, salvo que cuando pv>=length_v se retorna de la función.*

Para poder comprender el diagrama de flujo se facilita el significado de las variables utilizadas:

- Flag_y, flag_Cb son variables que indican si ya ha sido enviada toda la información referente a esas componentes.
- SOT/RfS son el el "start of transmission" y el "request for send" respectivamente.
- Py, pu son las variables que actúan como índices a los vectores de las componentes de luma y croma. Py_ant, pu_ant son los índices que guardan el valor anterior al último paquete enviado para utilizarlas en caso de un error de CRC.
- Length_y, length_u son los valores de la longitud de los vectores donde se guardan las componentes de la imagen.
- Vector_y, vector_u son los vectores que guardan la imagen codificada.

Normalmente, el receptor está en un estado inactivo de bajo consumo hasta que recibe un paquete con un SOT, lo que implica que se vaya a comenzar la transmisión de una imagen. A continuación, se replica con un ACK y se vuelve al estado inicial a la espera de la llegada de los datos. Cuando se detecta que ha llegado un paquete de información (por una cabecera especial), se pasa a leer el buffer de entrada. En caso de que el paquete sea el primero que transporte información de una componente, quiere decir que se envía la longitud del vector. Esto sirve para hacer la reserva de memoria apropiada, y la información que llegue en los sucesivos paquetes se almacenará en esa memoria.

Al final de cada paquete se comprueba si se ha llegado al final del vector y, si fuera el caso, se indica que ya ha sido transmitida esa componente de la imagen al completo (flag = 1). Cuando llegue el siguiente paquete se sabe por tanto que pertenece a la croma.

Una vez se tiene la información en el mismo formato que se envió, es necesario hacer el proceso inverso al de compresión. Esto implica deshacer la decodificación RLC, la ordenación en zig-zag, cuantización y finalmente aplicar la transformada inversa del coseno. Después de este proceso se tendría la imagen en formato YCbCr, por lo que habría que hacer la conversión a RGB.

4 EVALUACIÓN

Además de la funcionalidad, los principales parámetros que se desean verificar en esta sección son el consumo de potencia del dispositivo en todos los posibles estados y el rango de distancias en el que la comunicación es operativa. En este último aspecto, el fabricante establece que el dispositivo ha sido validado hasta los 10km pero no se indica en qué entorno ha sido realizada esta medición.

4.1 ESTUDIO DEL RADIO DE ACCIÓN DEL TRANSCEPTOR

Para llevar a cabo esta caracterización se ha procedido a situar los dos equipos en lugares con contacto visual pero lo suficientemente distantes como para alcanzar este radio de 10 km. El entorno elegido ha sido el de la bahía de Santander, ya que su característico cuello de botella permite que se puedan divisar lugares relativamente lejanos. Se han fijado mediciones con diferentes configuraciones de SF y BW para establecer diferentes niveles de sensibilidad, y comprobar cómo se comporta el enlace midiendo la cantidad de paquetes perdidos o con errores.

Figura 49. Mapa contenedor de los puntos de test utilizados.

A continuación se indican las coordenadas de cada uno de los puntos en los que se han realizado mediciones, además de la distancia en línea recta al equipo de referencia. Dicho equipo ha sido situado en las proximidades del Ayuntamiento de Camargo.

Lugar	Latitud	Longitud	Altitud (m)	Distancia(km)
Ayuntamiento Camargo	43º 25’ 23,13” N	3º 51’ 17,34” O	43	0
Polígono de Raos	43º 25’ 52,18” N	3º 49’ 35,56” O	0	2,22
Palacio de festivales	43º 27’ 44,93” N	3º 47’ 27,03” O	3	6,74
Somo	43º 27’ 17,39” N	3º 44’ 5,17” O	42	10,28

El transmisor va a emitir desde el punto de mayor altitud (Ayuntamiento de Camargo) y la recepción se realizará desde cada uno de los puntos indicados con el siguiente ratio de paquetes extraviados o con errores. Se han realizado medidas con diferentes configuraciones del Spreading Factor y de ancho de banda para comprobar la cantidad de paquetes perdidos/extraviados en función de la sensibilidad en el receptor, que tiene la siguiente expresión:

$$S = -174 + 10\ \log_{10} BW + NF + SNR \qquad \text{Eqn. 1}$$

El ancho de banda puede oscilar entre los siguientes valores: 125kHz, 250 kHz, 500 kHz. La sensibilidad se ve disminuida (en valor absoluto) en 3 dB cada vez que se incrementa el ancho de banda entre alguno de los valores anteriormente comentado. El SF oscila entre los siguientes valores: 7, 8, 9, 10, 11, 12. El aumento de una unidad en SF provoca una disminución de 2,5-3 dB (en valor absoluto) de la sensibilidad.

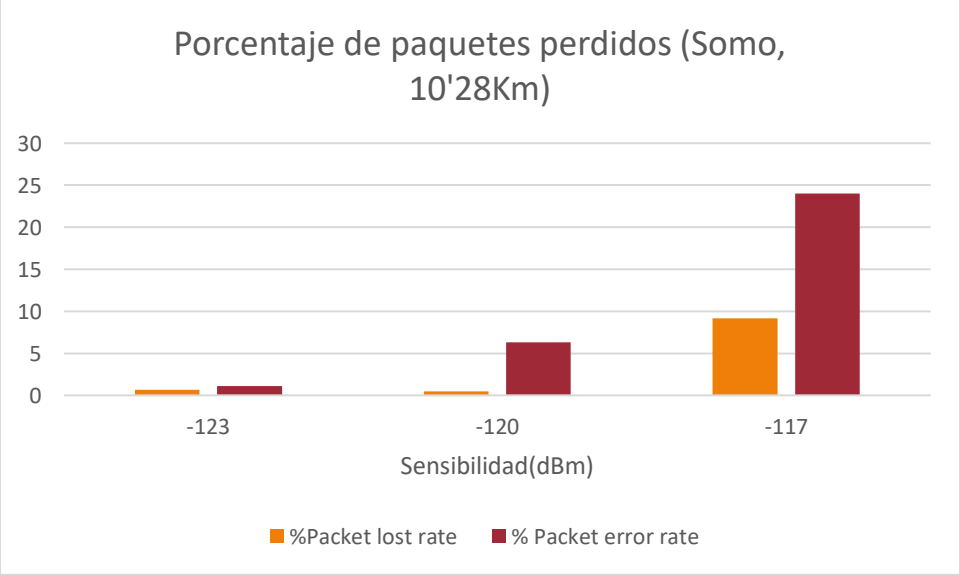


Figura 50. Tabla con el porcentaje de paquetes perdidos/con errores (distancia =10,28 Km, Somo) en función de la sensibilidad del receptor.

Para todas las sensibilidades posibles, las medidas realizadas en los emplazamientos que se situaban a una distancia menor a la decena de Km no presentaban problemas en cuanto a la

calidad del enlace. Por lo tanto, en los caso de Raos (2,2Km) y del Palacio de Festivales (6,74Km) el 100% de los paquetes son recibidos sin error para todas las sensibilidades.

En el punto que se encuentra más alejado (Somo, a 10,28Km) se observa cómo la cantidad de paquetes que no llegan a destino correctamente aumenta a medida que se reduce la sensibilidad. Es especialmente grave el aumento de paquetes con error y perdidos ocurrido cuando la sensibilidad en el receptor baja de los -120 dBm. Cabe destacar que esta tecnología permite la conectividad a distancias que superan la decena de kilómetros mientras que otras más instauradas como Bluetooth no supera el centenar de metros con la potencia para dispositivos clase 1 (20 dBm). Esto hace Lora sea una opción más atractiva tanto en redes domésticas (por su bajo consumo y mayor alcance) y para su conectividad en redes más amplias como las LPWAN.

Las dos únicas configuraciones de SF y BW que tienen como resultado una sensibilidad igual (o menor) a -120 dBm son BW=500KHz SF=7 y BW=250KHz SF=7. Estas deben ser evitadas en aplicaciones en las que sea requerida una alta fiabilidad a largas distancias.

4.2 CONSUMO DE LA PLATAFORMA LORA.

Si bien se puede consultar en la hoja de características el consumo del transceiver SX1276, este valor diferirá al conectarlo al módulo embebido por lo que se debe procede a hacer una medida del consumo de la plataforma tanto en estado de recepción como de transmisión. Para ello, se utiliza el medidor de potencia N6705, de Agilent Technologies.

Figura 51. Imagen del power analyzer N6705 de Agilent.

No se va a utilizar la parte de la entrada USB, ya que dispararía el consumo, sino que la salida del analizador se va a conectar a los pines 24 (Vin) y 22(GND) del CN7 con una tensión de 9V y una corriente de 450mA. El módulo Lora procede a enviar paquetes de ping esperando la respuesta por parte del segundo módulo, mostrando la siguiente gráfica de corriente consumida.

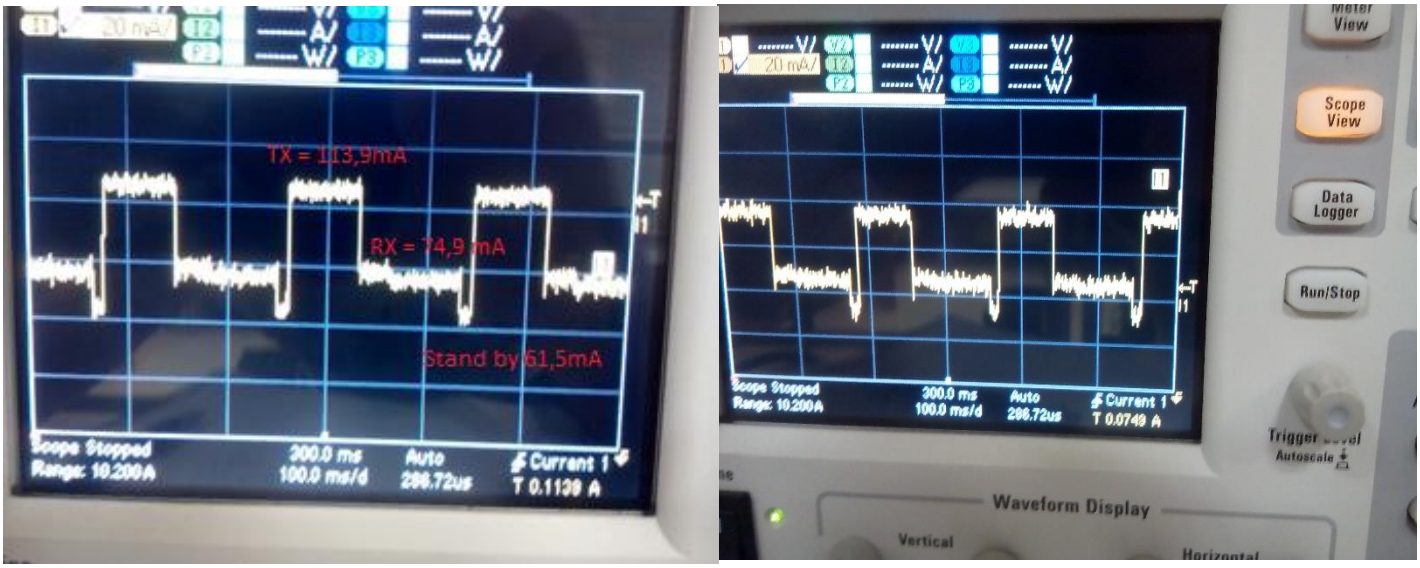


Figura 52. Forma de onda de la corriente en función del tiempo para un ciclo de transmisión y recepción con Pout=20dBm.

La potencia de salida a la cual se ha realizado esta primera medida es a 20 dBm con un resultado de 113,9mA en transmisión y de 74,9mA en recepción. También se realizan medidas para potencias de salida de 17dBm, 13dBm, 7dBm. En la siguiente tabla se muestran los consumos para diferentes potencias durante la transmisión.

Potencia de salida(dBm)	Plataforma Lora(mA)	SX1276(mA)
Todas	65	11
17	102,1	87
13	99,8	29
7	93,8	20

Una de las desventajas de la plataforma Lora es que al estar acoplado el transceiver a un ARM cortex M4, la bajada de potencia consumida conforme a la potencia de salida no es tan acentuada como los datos proporcionada en la hoja de datos del transceiver. En cuanto a la recepción, los datos son mucho más estables y oscilan en torno a los 75mA. El transceiver tiene un consumo estable en función de la potencia de salida de 11mA.

4.3 EVALUACIÓN DEL COSTE POR UNIDAD FABRICADA

En un mundo globalizado y cada vez más competitivo es de vital importancia conocer el coste de producción de cada una de las unidades diseñadas. Para ello, es vital saber discernir los costes recurrentes de los que no lo son y añadir el factor del volumen, es decir, la cantidad de piezas a fabricar. Los costes recurrentes se definen como los costos repetitivos en una empresa al fabricar de manera repetitiva un producto, siendo los no recursivos aquellos que no se requerirán para la producción en el siguiente año. Se llega, por tanto, a una expresión del coste por unidad que tiene los siguientes factores:

$$C_P = \frac{C_{NRE}}{V} + C_R$$

Figura 53. Fórmula del coste de producción por unidad [17].

Donde:

- C_{NRE} son los costes no recurrentes.
- V es el volumen de producción.
- C_R son los costes recurrentes.

Los costes no recurrentes engloban las siguientes tareas:

- Diseño del producto.
- Evaluación.
- Costes de certificación.

En este apartado, el principal coste son los gastos de recursos humanos asociados al desarrollo del software que implementa la funcionalidad, además del coste de la verificación de que el componente cumple las directivas necesarias para poder obtener el marcado CE, lo cual es imprescindible para su comercialización en la Unión Europea. También se deben computar el de los materiales técnicos utilizados para poder desempeñar la tarea (PC, osciloscopios, polímetros,…).

El tiempo de desarrollo se va a fijar en dos meses (44 días laborables) con una jornada laboral de 8 horas diarias, siendo el coste del trabajador por hora de 30€ (coste de empresa). Coste de personal (C_H):

$$C_H = 30 * 8 * 44 = 10560€$$

Además, hay que imputar todos los gastos asociados al material del que debe disponer el asalariado para poder realizar su trabajo. Para hacer este cálculo se va a utilizar el período de amortización.

Herramienta	Período de amortización (años)	Coste	Tiempo de utilización
PC	2	1100€	2 meses
Osciloscopio Metrix OX803BS	6	969€	1 semana
Analizador de potencia N6705 Agilent	6	7765$	1 semana

La siguiente expresión muestra el coste por la utilización de cada material (C_M) durante el tiempo requerido:

$$C_M = \frac{Coste}{t_{amortización}} * t_{utilización}$$

Figura 54. Fórmula del coste asociado al material empleado.

En cuanto al marcado CE necesario para ofrecer el módulo Lora, se declara que está conforme a las siguientes regulaciones de distintos organismos:

- EN 301 489-1(1.9.2)- (1.6.1) EN 61000-3.2:2006. Electromagnetic Compatibility (EMC) Parte 3.2 se aplica a emisiones de corriente armónicas. Corriente de entrada al equipo menor a 16 A.

  No es necesaria la parte 3.3 referente a la limitación en cambios de tensión para sistemas conectados a sistemas de alimentación públicos de baja tensión ya que se utilizará una pila de 9V.

- EN 55022:2010. Information Techonology Equipment. Radio Disturbance Characteristics. Limits and methods of measurement.
- EN 60950-1: 2006 Information Technology Equipment. Part 1: General Requirements.

Estas reglas establecen las normas en cuanto a los límites legales tanto de radiación electromagnética (EMC) conducida, como radiada. Hay que aclarar que un dispositivo es electromagnéticamente compatible con su entorno si cumple cada una de las siguientes condiciones:

- No causa interferencia a otros sistemas.
- No es susceptible a las emisiones de otros equipos.
- No se causa interferencias a sí mismos.

En la siguiente tabla se muestran los costes de material asociados a cada uno de los componentes y el coste total.

Descripción	Coste
PC	91,67€
Osciloscopio Metrix OX803BS	3,65€
Analizador de potencia N6705 Agilent	26,96$
Total(C_M)	120,36€
C_H	10560€
Coste de certificación	12009,00€
C_{NRE}	**22809,72€**

Se ha aplicado una equivalencia de 1,0765$/€.

Los costes no recurrentes se pueden calcular como $C_{NRE}=C_H+ C_M+$ Coste de certificación.

En los costes recurrentes se van a computar el coste de adquisición del transceiver Lora SX1276 y del microprocesador STM32F401-RE.

Dispositivo	Precio
Nucleo 64 STM32F401-RE	12,74$
Transceiver SX1276	59,25€
Pila 9V	1,50€
Caja (12 x 6,5 x3,98 cm)	3,876€
C_R	**76,460€**

Se ha aplicado una equivalencia de 1,0765$/€.

La única variable que queda por determinar es el volumen de producción, en base a lo cual se determina el precio final de producción por unidad. La siguiente gráfica muestra ese precio en función de la variable V.

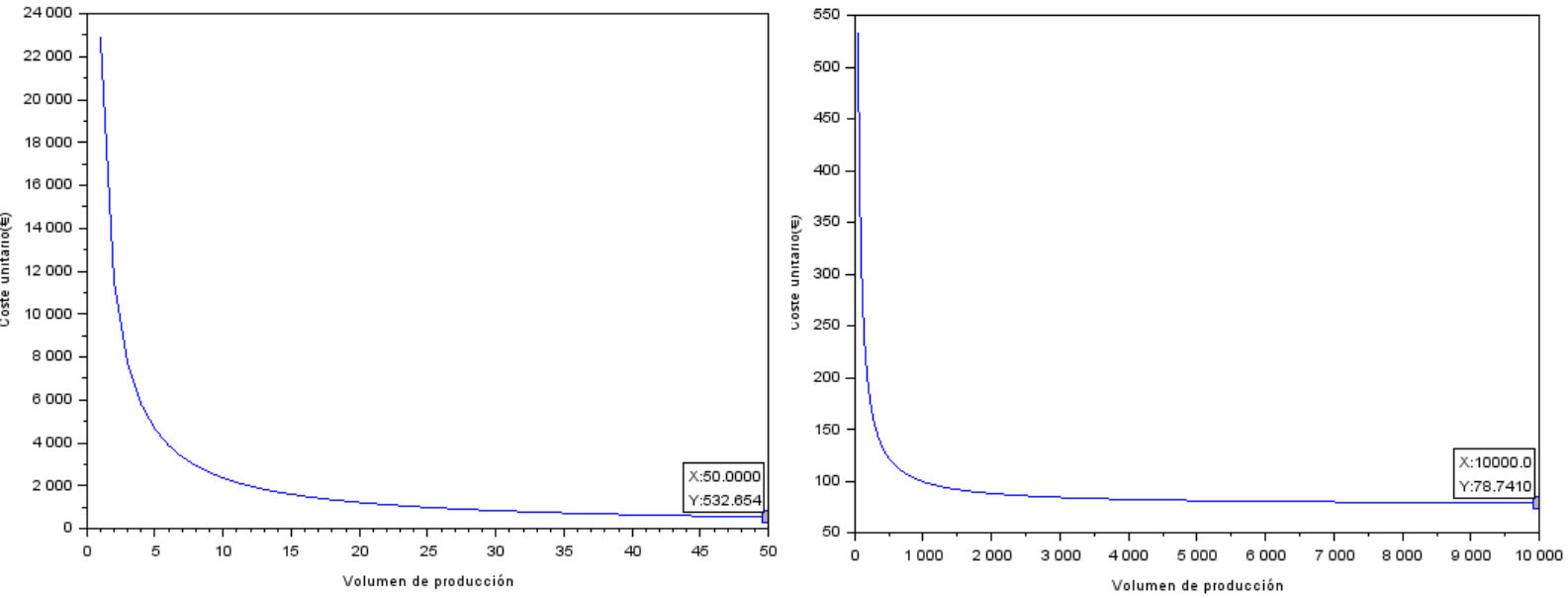


Figura 55. Fórmula del coste asociado al material empleado.

Las dos curvas muestran los valores del coste unitario para diferentes rangos de V (0-50, 50-1000). Como es lógico, para un volumen de pocas unidades el coste es muy elevado, pero a medida que se incrementa esta variable se tiende a que el miembro que cobre mayor importancia sea el de los costes recurrentes. Para un volumen de 10000 unidades ya se está tan solo 2€ por encima de los costes recurrentes y a partir de las 6500 el coste unitario está por debajo de los 80€.

Para reducir los costes recurrentes se podría abordar la posibilidad de diseñar un SoC capaz de realizar el procesado asociado al compresor JPEG para a su vez comunicarse con el transceptor SX1276. Se eliminarían de esta forma los 12,74$ asociados a la Nucleo board en los costes recurrentes, pero se deberían utilizar más recursos de diseño que engordarían la cantidad de los no recurrentes, por lo que es una decisión empresarial de alta responsabilidad.

Se prevee que para que el IoT termine de despegar, es necesario que llegue a un coste relativo al hardware por debajo de 1 dólar, por lo que todavía queda un amplio camino por recorrer en ese sentido.

5 CONCLUSIONES

En este proyecto se ha abordado el diseño e implementación de un sistema de compresión y transmisión de imágenes mediante la tecnología Lora. Como resultado se ha conseguido con éxito la transmisión de una imagen a distancias superiores a la decena de Kilómetros (siempre que haya contacto visual) con potencias del orden del 20dBm (potencia de un Bluetooth clase 1).

Desde un punto de vista hardware, el sistema utiliza un sistema embebido basado en un ARM de la familia STM32 y un transceiver Lora. La imagen se transmite desde un PC auxiliar al sistema, el cual se alimenta normalmente con la conexión USB, aunque también se ha verificado su alimentación mediante una pila de 9V.

En cuanto al compresor de imágenes, se ha implementado un codificador que permite modificar diversos parámetros, como el sub-muestreo y las tablas de cuantificación. Por ejemplo, que se ha observado un límite del ratio de compresión para imágenes con alta redundancia espacial, que se ha situado en un factor de 16. Para las fotos con alto nivel de detalle se observó un comportamiento lineal de la compresión, proporcional a las tablas de cuantización. Para un factor alto (x8), ya se empieza a detectar distorsión de la imagen, por lo que no es recomendable superar dicho umbral.

Además se ha desarrollado un protocolo de comunicaciones que permite gestionar no solo la comunicación entre el PC que almacena la imagen y el sistema embebido, sino también la transmisión de la imagen por radio. El protocolo implementado permite gestionar errores de transmisión y problemas de enlace.

Uno de los objetivos del proyecto era evaluar la capacidad del transceptor con modulación Lora para permitir la comunicación a largas distancias (10Km) utilizando una potencia de transmisión mínima. Los resultados han sido aceptables ya que se ha conseguido establecer enlaces en las distancias esperadas, pero aún queda un largo camino que andar para poder aumentar la tasa de transmisión de bits. Esto supondrá la eclosión total de esta tecnología ya que se podrá utilizar en un rango incontable de nuevas aplicaciones, además de poder aumentar la comunicación entre los dispositivos desplegados en una red LPWAN ya existente.

De alguna manera este proyecto puede utilizarse para otras aplicaciones relacionadas con el IoT, ya que el código utilizado para establecer el radioenlace es absolutamente reutilizable y legible. Dicho código ha sido documentado en este proyecto mediante el uso de máquinas de estado.

Por último, el proyecto también evalúa el consumo y coste de la solución diseñada, que se encuentran dentro de los valores esperado.

5.1 LÍNEAS FUTURAS DE TRABAJO

Siempre que se finaliza un proyecto se puede iniciar otro que lo mejore y perfecciones, y esta no es una excepción:

- Utilización de las nuevas plataformas multinúcleo para poder paralelizar el procesado y reducir el tiempo necesario para enviar una imagen y poder aplicar restricciones temporales.
- Emplear el compresor de imagen como embrión para un compresor de vídeo, añadiendo toda la parte necesaria para cálculo de “Motion Vectors”. Juntando estas dos ideas se podría llegar a pensar en desplegar una red para broadcasting de vídeo cuando esta tecnología llegue a los data rates mínimos.
- Abordar la otra parte de la comunicación de una red LPWAN, es decir, las conexiones IP estándar entre las puertas de acceso (gateways) y los servidores. Así mismo se podrían introducir alguna funcionalidad relacionada con el “big data” ya que puede ser interesante el procesado de grandes cantidades de datos procedentes de diferentes sensores.
- Implementación de una conexión directa Wifi o bluetooth entre el usuario (a través de un Smartphone) y el transceiver para tener acceso a las lecturas que ha hecho un determinado sensor.
- Descubrir nuevos rangos de frecuencia donde haya algún resultado positivo en cuanto a distancias cubiertas satisfactoriamente y reducción de interferencias.

6 REFERENCIAS

[1] Internet of Things: The Complete Reimaginative Force – *TCS Global Trend Study (July 2015).*

[2] Video coding for low bit rate communication. Appendix III: Exampples for H.263 encoder/decoder implementations – *ITU-T Recommendation H.263.*

[3] http://www.equasys.de/colorconversion.html

[4] https://www.wikipedia.org

[5] Ramon de Klein, "Instructions of use for a serial communicacion API", *Hardware programming (2003).*

[6] Datasheet del transceptor SX1276: http://www.semtech.com/images/datasheet/sx1276.pdf

[7] Lora Modem designer's guide transceptor SX1276: http://www.semtech.com/images/datasheet/LoraDesignGuide_STD.pdf

[8] Pablo Sánchez Espeso, "Imagen y vídeo: Compresión JPEG", *Sistemas Electrónicos Multimedia Parte 3.4.*

[9] IBM Long Range Signaling and Control (LRSC) – *A highly efficient M2M communication.*

http://www.research.ibm.com/labs/zurich/ics/lrsc/lmic.html

[10] The Ultimate Long Range Wireless Solution – *Lora World.*

https://www.lora-alliance.org/What-Is-LoRa/Technology

[11]Lora modem designer's guide: http://www.semtech.com/images/datasheet/LoraDesignGuide_STD.pdf

[12]Lora modulation Basics guide: http://www.semtech.com/images/datasheet/an1200.22.pdf

[13] Thomas Telkamp, "*Lora crash course", The Things Network (2016)*

https://www.youtube.com/watch?v=T3dGLqZrjIQ&t=2702s

[14] STM32 Nucleo-64 boards User manual: http://www.st.com/content/ccc/resource/technical/document/user_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/translations/en.DM00105823.pdf

[15] http://www.sigfox.com

[16] Lora vs LTE-M vs Sigfox, -*why the IoT is at risk of being stillborn.*

http://www.nickhunn.com/wp-content/uploads/downloads/2015/12/LoRa-vs-LTE-M-vs-SigFox.pdf