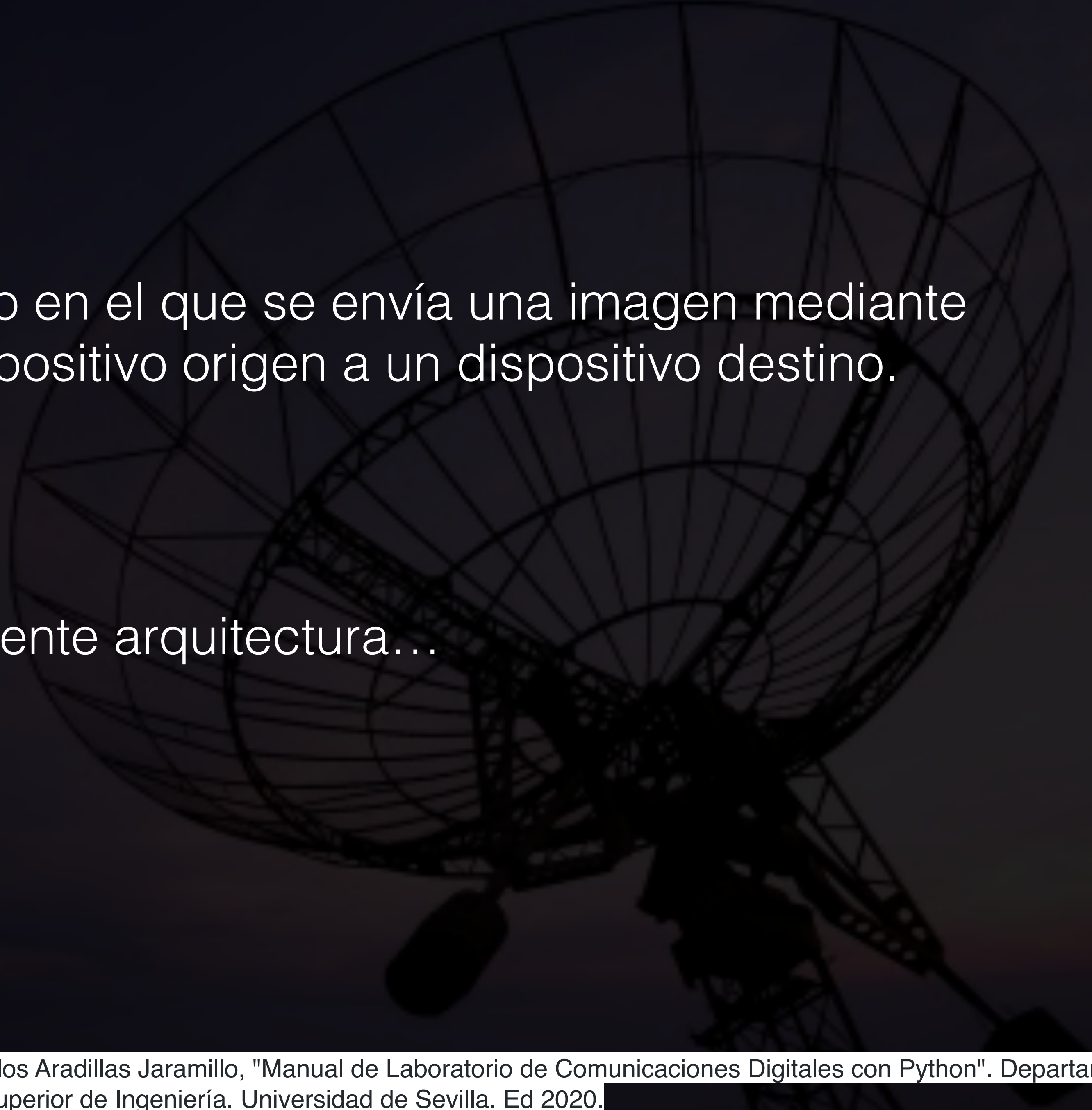


# Ejemplo de aplicación de las prácticas de Comunicaciones Digitales. Parte I

Comunicaciones Digitales - 3º GITT - Universidad de Sevilla

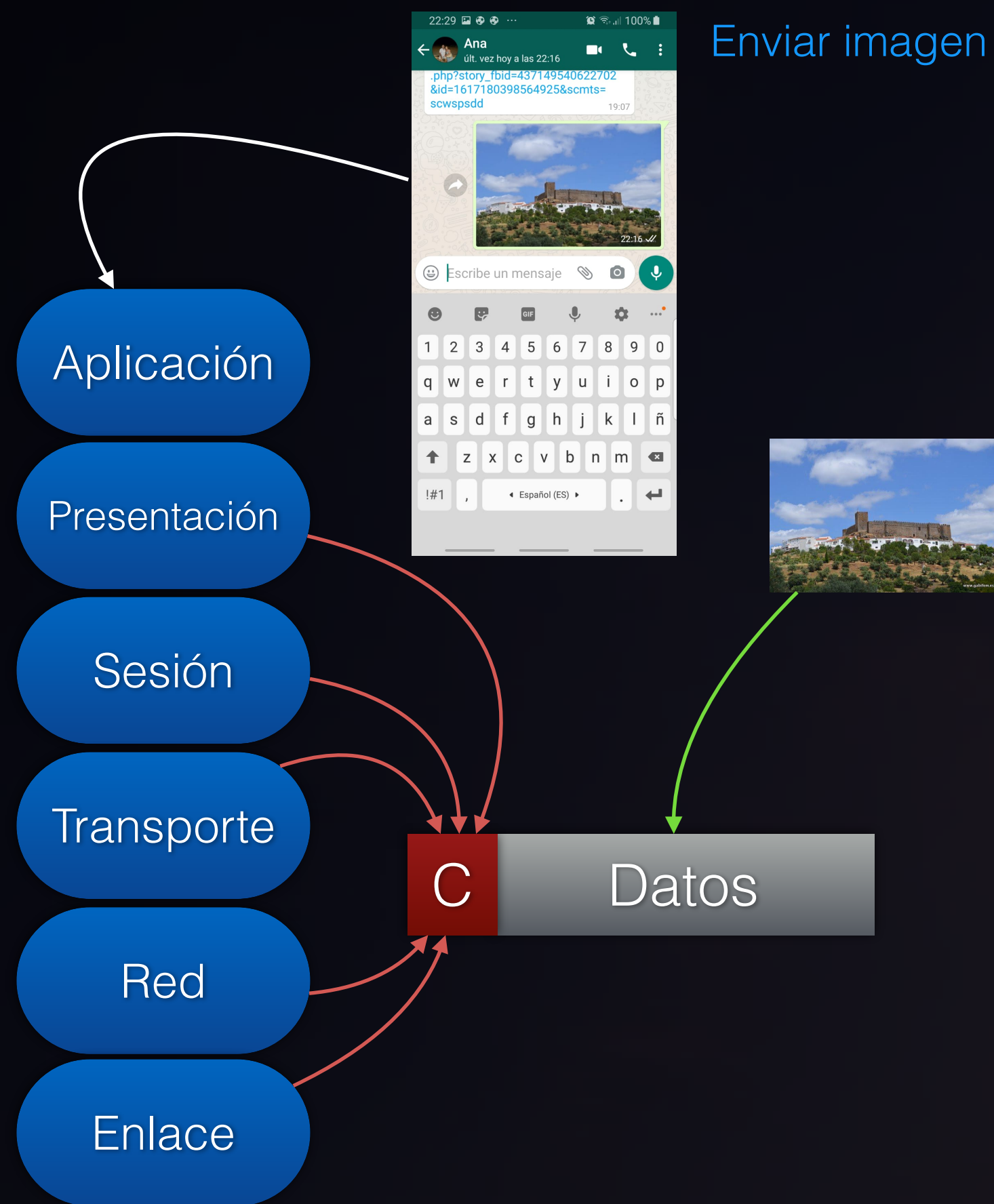
José Carlos Aradillas Jaramillo



Vamos a emular un escenario en el que se envía una imagen mediante una aplicación desde un dispositivo origen a un dispositivo destino.

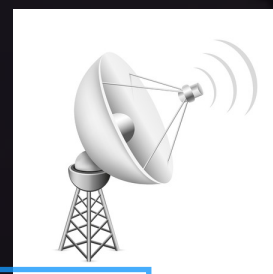
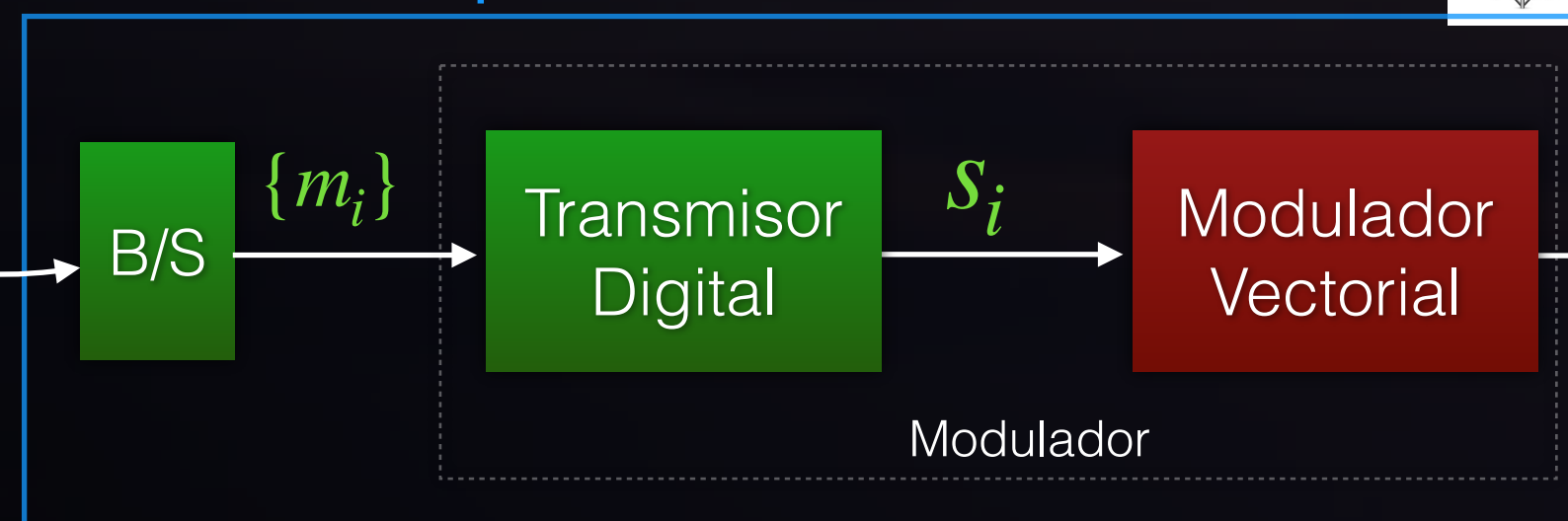
Para ello planteamos la siguiente arquitectura...



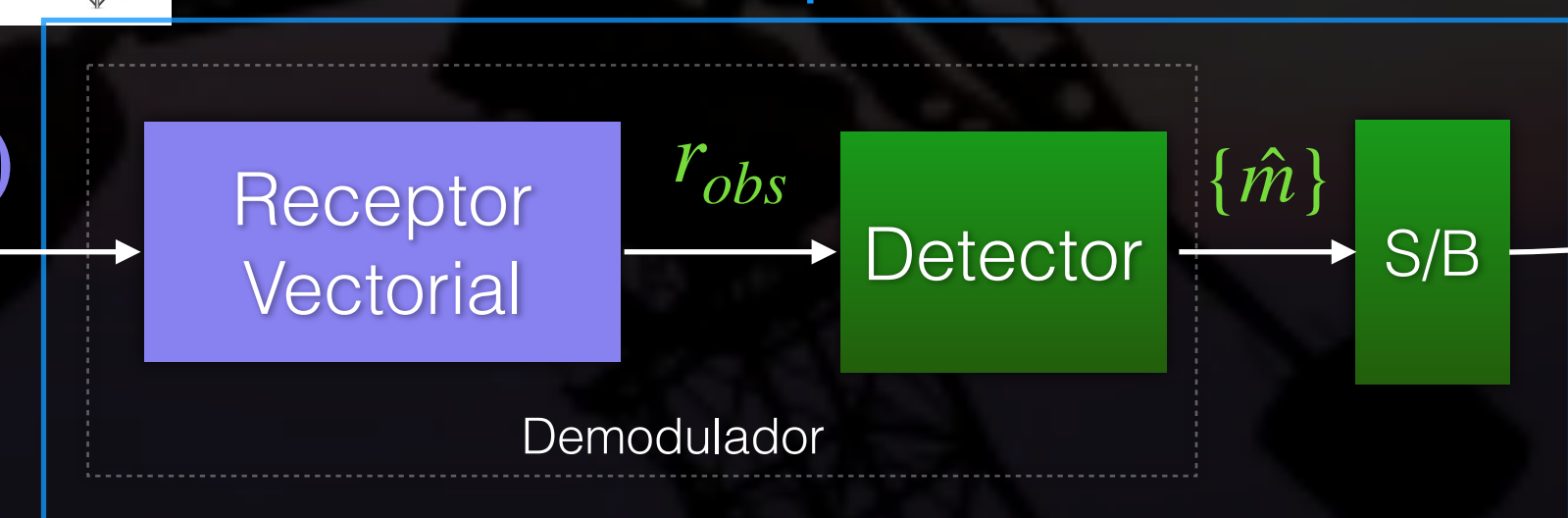
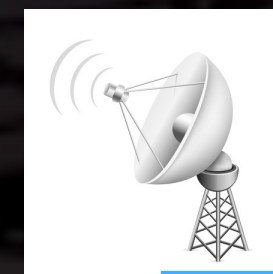


$$B_n = [1,0,0,1,1,1,0,0,1,1,0,...]$$

Capa física



Canal



Capa física

$$\hat{B}_n = [1,0,0,1,1,1,0,0,1,1,0,...]$$



# Dividimos el escenario en 3 bloques

1. El lado completo del usuario transmisor junto con el canal ya implementado.

Se obtiene la señal  $R(t)$  mediante la función `load_signal()`

2. El bloque del receptor vectorial. **No está implementado.**  
**Es lo único que hay que implementar.**

Obtener el vector de observación “ $r$ ” a partir de  $R(t)$ .  
Hay que tener en cuenta los parámetros del transmisor.  
También se debe obtener el alfabeto.

3. Implementación del lado completo del usuario receptor excepto el receptor vectorial.

A partir del vector de observación “ $r$ ” obtenido en el bloque 2 y el alfabeto (constelación) se obtiene la imagen recibida en la aplicación, mediante la función `detecta_y_procesa()`







# Dividimos el escenario en 3 bloques

1. El lado completo del usuario transmisor junto con el canal ya implementado.

Se obtiene la señal  $R(t)$  mediante la función `load_signal()`

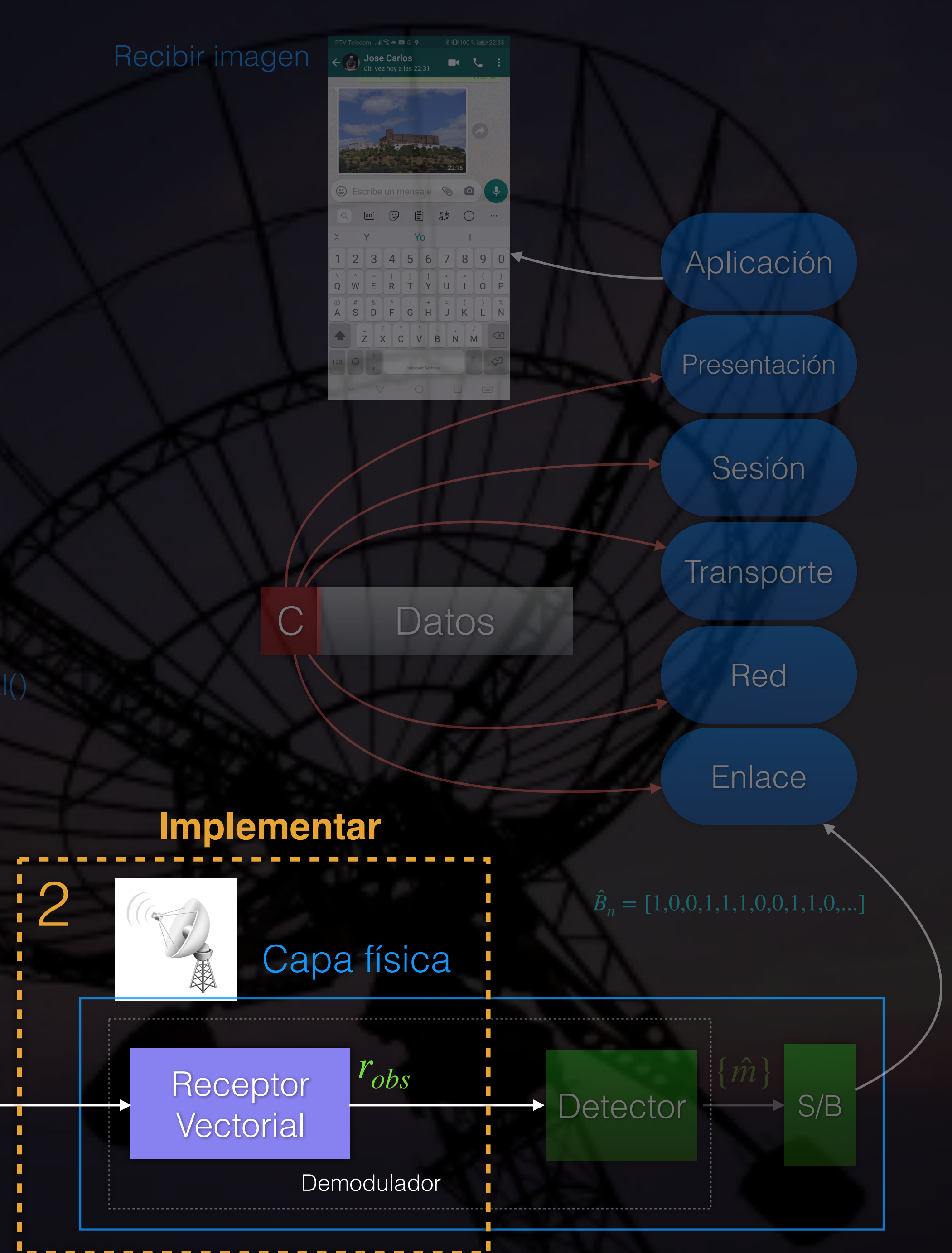
2. El bloque del receptor vectorial. **No está implementado. Es lo único que hay que implementar.**

Obtener el vector de observación “ $r$ ” a partir de  $R(t)$ .  
Hay que tener en cuenta los parámetros del transmisor.  
También se debe obtener el alfabeto.

3. Implementación del lado completo del usuario receptor excepto el receptor vectorial.

A partir del vector de observación “ $r$ ” obtenido en el bloque 2 y el alfabeto (constelación) se obtiene la imagen recibida en la aplicación, mediante la función `detecta_y_procesa()`







# Dividimos el escenario en 3 bloques

1. El lado completo del usuario transmisor junto con el canal ya implementado.

Se obtiene la señal  $R(t)$  mediante la función `load_signal()`

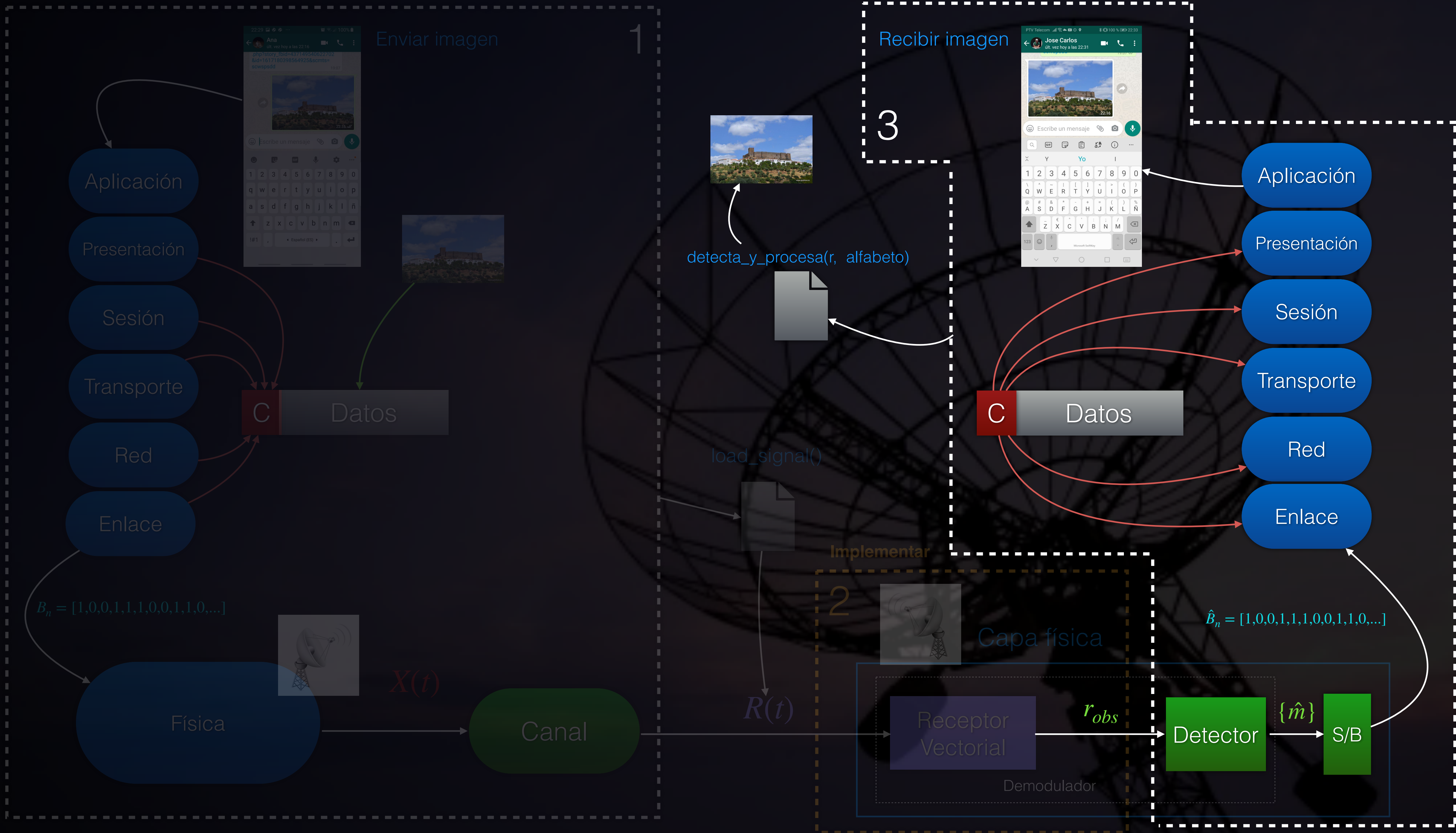
2. El bloque del receptor vectorial. **No está implementado. Es lo único que hay que implementar.**

Obtener el vector de observación " $r$ " a partir de  $R(t)$ .  
Hay que tener en cuenta los parámetros del transmisor.  
También se debe obtener el alfabeto.

3. Implementación del lado completo del usuario receptor excepto el receptor vectorial.

A partir del vector de observación " $r$ " obtenido en el bloque 2 y el alfabeto (constelación) se obtiene la imagen recibida en la aplicación, mediante la función `detecta_y_procesa()`







# Se proporcionan tres archivos:

- EjercicioExtraParte1.py —> Fichero principal, en el mismo se carga la señal  $R(t)$ , se demodula y se envía a las capas superiores almacenando la imagen resultado en un fichero .jpg dentro del mismo directorio. Hay que **implementar el demodulador** donde se indica.
- utils.py —> contiene las funciones **load\_signal()**, y **detecta\_y\_procesa()** con la implementación de los bloques 1 y 3 respectivamente. **¡NO EDITAR ESTE FICHERO!**
- Rt1.npy —> Fichero .npy que contiene las muestras de la señal recibida  $R(t)$  en el dominio temporal a la salida de un canal determinado. No hay que hacer nada con este fichero, ya está implementada su carga en el fichero principal.
- Rt2.npy —> Fichero .npy que contiene las muestras de la señal recibida  $R(t)$  en el dominio temporal a la salida de un canal diferente al anterior. Cuando se resuelva el ejercicio con la demodulación de la señal en Rt1.py, cambiar en el fichero principal la línea donde se carga la señal para que cargue Rt2.py.



La señal recibida  $R(t)$  se corresponde con la salida de un transmisor 32-PAM **paso de banda** tras sufrir los efectos de un canal AWGN. Los parámetros del transmisor son los siguientes:

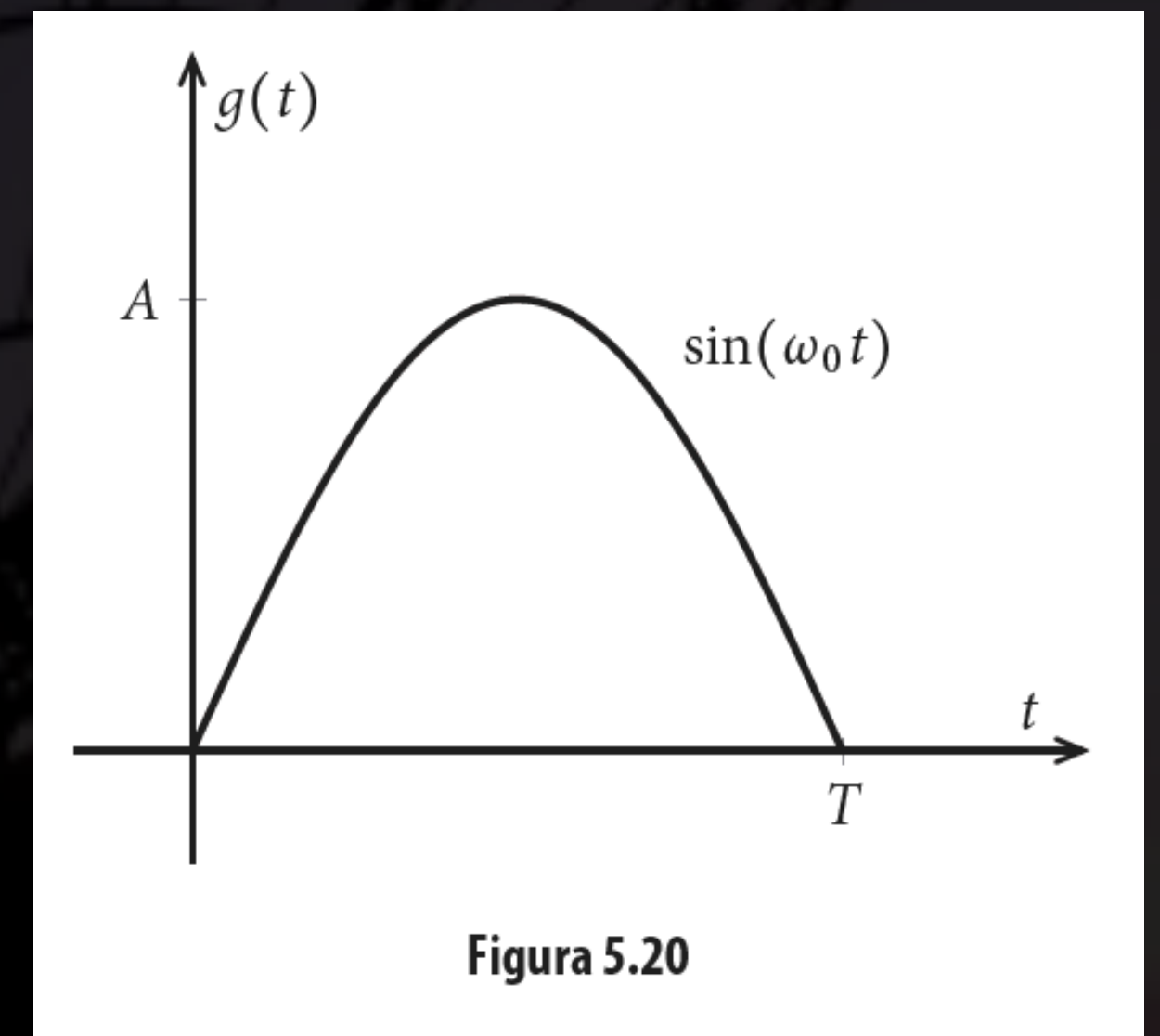
Energía promedio de bit  $\rightarrow E_b = 5$  Julios

Número de muestras por símbolo  $\rightarrow L = 64$

Tasa de transmisión de bit  $\rightarrow R_b = 300$  Mbps

Frecuencia de la portadora  $\rightarrow \omega_c = 16\pi/T$

Pulso conformador en banda base mostrado en la Figura 5.20



# Resumiendo...

1. Implementar el bloque 2 con ayuda de las herramientas aprendidas en la práctica 6 para demodular la señal Rt1.py
2. Cambiar la señal de entrada e incluir la del fichero Rt2.py

¿Qué diferencias hay entre ambos escenarios?

Enviar la imagen recibida (.jpg) en ambos escenarios y el código a la dirección de correo electrónico [jaradillas@us.es](mailto:jaradillas@us.es)