Escuela Técnica Superior de Ingenieros
de Telecomunicación

Universidad Politécnica de Madrid

Sistema digital de un radar CW-LFM en banda S

4 de febrero de 2019

Autor:
Luis Alberto Gómez

Índice

1. Funcionamiento general

El sistema usa un microcontrolador del fabricante *STMicroelectronics*, en concreto el modelo STM32F205VET6. La placa fabricada dispone de dos conectores USB (Figura 1). Uno de ellos sólo proporciona alimentación a los circuitos, mientras que el otro, además de alimentar, también permite la transmisión de información. En el caso de que un solo USB no sea suficiente para alimentar el sistema se pueden conectar ambos a la vez **(se debe tener en cuenta que ambos USB deberían estar conectados al mismo ordenador)**. También dispone de un botón para resetear el sistema.

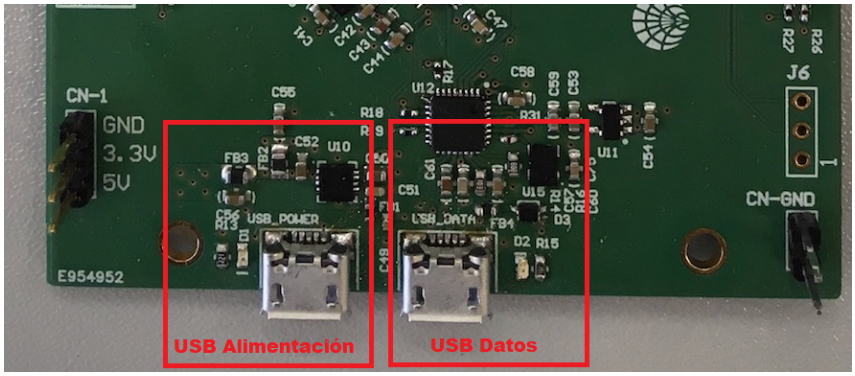

Figura 1: Conectores USB.

El sistema realiza tres funciones básicas:

- Adquisición de señal. Se utiliza un conversor analógico digital (ADC) que dispone de una entrada. Muestrea la señal y la digitaliza. Tiene una resolución de 12 bits, una frecuencia de muestreo máxima de 2 MHz y las tensiones que admite de entrada van de 0 a 3,3 V.
- Generación de señal. Se utiliza un conversor digital analógico (DAC) para generar la señal de modulación del VCO. Se puede fijar un valor de tensión concreto (y por tanto una frecuencia fija) o realizar un barrido en tensión. Puede generar tensiones entre 0 y 3,3 V y tiene una resolución de 12 bits.
- Envío de los datos adquiridos a un ordenador. Los datos digitalizados se envían en bloques de la longitud del tiempo de rampa[1]. Cada valor tiene un tamaño de 16 bits. Como el ADC tiene una resolución de 12 bits, los 4 bits más significativos se desaprovechan. Los valores enviados tienen un rango de 0 a 4095, correspondiendo a 0 V y 3,3 V respectivamente.

En la Figura 2 se puede observar el diagrama de flujo del sistema. Una vez iniciado, de forma sincronizada se genera la señal deseada que modula el VCO y se captura la señal recibida con el ADC. La cantidad de puntos digitalizados depende del tiempo de rampa, ya que la frecuencia de muestreo está fijada a 2 MHz. Este bloque de puntos se envían utilizando el puerto serie a un ordenador, y para recibirlos se puede usar cualquier programa capaz de leer un puerto serie, como MATLAB. Una vez iniciado el sistema, tanto la generación de señal como la adquisición de datos se realiza de forma continua.

[1] Al realizar la programación del sistema se ha elegido este método por su sencillez.

En la Figura 3 se puede observar un diagrama temporal de la generación y adquisición de señal (el sistema está en su configuración por defecto). En la parte superior se puede observar una señal de depuración disponible en el pin 2 del conector identificado como "J6" en la placa. Esta señal cuadrada cambia de nivel cada vez que se empieza a adquirir un nuevo bloque de puntos con el ADC. En la parte inferior de la Figura 3 se puede ver las rampas generadas por el DAC. Como se puede ver la generación de las rampas y la digitalización están sincronizadas.

Creo que esto habria que pasarlo a otra seccion dependeinte de la version.

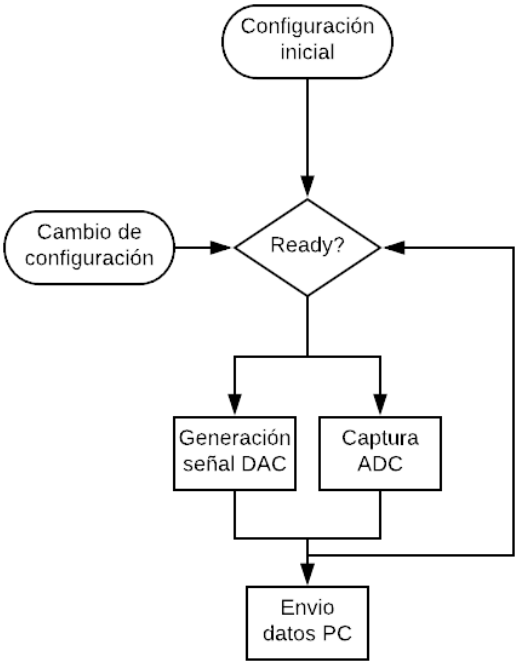

Figura 2: Diagrama de flujo del sistema.

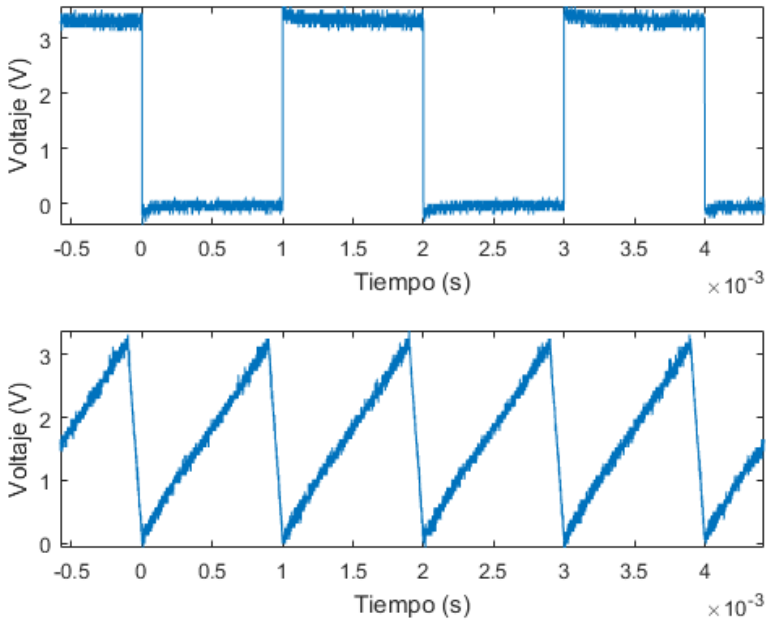

Figura 3: Captura del ADC y generación de señal.

Al iniciar el sistema se realiza una configuración inicial que fija los valores por defecto del tiempo de rampa (1 ms) y su voltaje mínimo y máximo (de 120 mV a 3,22 V). Se empieza a generar señal y adquirir. Sin embargo, el VCO se mantiene desactivado hasta que el usuario lo active, por lo que por defecto no se genera señal de RF.

Revisar

En necesario destacar que en el caso de usar el sistema operativo Windows es necesario instalar un driver (proporcionado junto a este documento) para un funcionamiento correcto del sistema. En el caso de usar una distribución Linux o MacOS no es necesario.

Añadir en el git

2. Configuración

2.1. Primera version

El sistema tiene varios parámetros que se pueden configurar a través de diferentes comandos. Cada comando corresponde con una cadena de caracteres específica. Para realizar una configuración hay que enviar la cadena de caracteres correspondiente a través del puerto serie. En la sección 3 se puede ver algunos ejemplos desarrollados en Matlab. A continuación se detallan las posibles configuraciones:

- Ganancia del amplificador de baja frecuencia **(en caso de utilizar la configuración del sumador ya no sería configurable)**. Se pueden fijar ganancias de 3 a 50 en unidades naturales. Para ello se utiliza el siguiente comando, siendo "X" el valor que se quiere fijar. El valor por defecto es 4.

  ```
  "setgain X\n"
  ```

- Habilitar o deshabilitar el VCO. Se utiliza el comando descrito a continuación. Cuando el valor de "X" vale "1" el VCO genera señal y si vale "0" para de generar. El sistema se inicia con el VCO desactivado.

  ```
  "envco X\n"
  ```

- Tiempo de rampa. Se puede variar entre 100 μs y 4 ms. Se genera una rampa lineal en tensión con el nuevo tiempo de rampa, además, se fija un tiempo de caída correspondiente al 10 % del tiempo de rampa. Para ello se utiliza el comando descrito a continuación. "time" es el valor del tiempo elegido en μs. Por defecto, el tiempo de rampa es 1 ms. Sobrescribe cualquier rampa previamente definida. Mantiene los voltajes mínimo y máximos fijados por el comando "setrampvoltages" que se explica a continuación.

  ```
  "setramplength time\n"
  ```

- Amplitud de tensión de la rampa. Se puede configurar el voltaje mínimo y máximo de la rampa, con valores entre 0 y 4095, que corresponden con 0 y 3,3 V. Genera una rampa lineal en voltaje con los límites dados. Si se quisiese fijar otro tipo de barrido se utilizaría la configuración de una rampa definida por el usuario. El comando utilizado es el siguiente, siendo "v1" la tensión mínima y "v2" la máxima. Por defecto los voltajes son: v1 = 150 y v2 = 4000. Sobrescribe cualquier rampa previamente definida. Usa el tiempo de rampa definido con el comando "setramplength".

  ```
  "setrampvoltages v1 v2\n"
  ```

- Configuración de voltaje a la entrada del VCO. Se puede seleccionar una frecuencia de oscilación fija entre los límites que proporciona el VCO. Para ello se fija una tensión específica utilizando el siguiente comando. El DAC esta alimentado entre 0 y 3,3 V y tiene 12 bits de resolución. Por tanto el valor de "voltage" va de 0 a 4095.

  ```
  "setvoltage voltage\n"
  ```

- Configuración de una rampa de modulación definida por el usuario. Es posible configurar una forma de rampa arbitraria. Para hacer esto se ha desarrollado una función de Matlab (llamada "set_ramp.m"), que se puede ver en la sección 3. Esta función tiene como parámetros un objeto de tipo "serial", un vector ("points") con los puntos de la rampa y la longitud de la rampa ("len") deseada en μs. El número de puntos del vector debe corresponder con la longitud de la rampa. Una vez fijada una rampa se mantiene hasta que se vuelva a realizar un cambio en la configuración.

  ```
  set_ramp( serial, points, len)
  ```

3. Ejemplos de uso

Se han desarrollado algunos "scripts" de Matlab para demostrar el uso de varias funciones.
A continuación se puede ver un pequeño "script" que el abre un puerto serie, activa el VCO, fija una ganancia de 4 y recibe 2000 puntos. Esta cantidad de puntos leída se puede variar por el usuario. En este ejemplo se usa 2000 ya que sería la cantidad de puntos correspondientes a un tiempo de 1 ms, que dada la configuración por defecto, es el tiempo correspondiente a una rampa. Los valores del resto de parámetros se mantendría por defecto (1 ms de tiempo de rampa y voltajes de rampa entre 0,12 V y 3,22 V). El nombre del puerto serie dependerá del sistema operativo usado.

```
delete(instrfindall)
ser = serial('COM10', 'InputBufferSize', 10000);
fclose(ser) % para evitar problemas primero se cierra
fopen(ser) % y despues volvemos a abrirlo
fprintf(ser, 'envco 1\n'); % habilitar el VCO
fprintf(ser, 'setgain 4\n'); %fija la ganancia
data = fread(ser, 2000, 'uint16'); % leer 2000 valores
```

El siguiente "script" genera un ejemplo de rampa definida por el usuario y realiza la configuración haciendo uso de la función "set_ramp.m". Se ha fijado un tiempo de caída del 10 % del tiempo de rampa. La rampa generada se puede ver en la Figura 4.

```
clear, close all
Tc = 1000; %microsegundos (Tiempo total de rampa)
caida = 100; %microsegundos
subida = linspace(150, 4000, Tc-caida); %Voltage de 150 a 4000.
bajada = linspace(4000, 150, caida);
rampa = floor([subida bajada]);
plot(rampa*3.3/4096),xlabel('Tiempo (\mus)'), ylabel('Voltaje (V)')
%%
delete(instrfindall)
serial = serial('COM10');
fclose(serial);
fopen(serial);
set_ramp(serial, rampa, Tc); %Enviar configuracion de la rampa
fclose(serial);
```

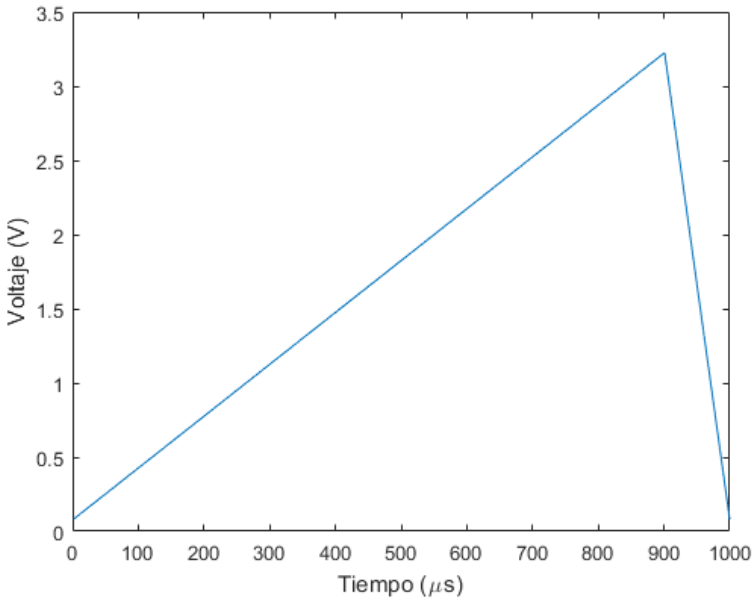

Figura 4: Rampa lineal en voltaje.

Para evitar perder datos se ha comprobado que la forma óptima es leer y escribir en un archivo los datos de forma continua. Al no estar realizando ningún procesado de los datos la lectura es lo suficientemente rápida como para evitar que se llene el buffer de recepción del ordenador y se pierdan bloques de datos. Después se realizaría el procesado de los datos guardados en el archivo. En anexos se puede ver un ejemplo en Matlab de como hacer esto sin perdida de datos.

4. Anexos

La función "set_ramp.m" es la siguiente:

```
function [  ] = set_ramp( serial, points, len)
% Los parametros de la funcion son el puerto serial a utilizar
% un vector con los puntos deseados en el rango de 0 a 2^12-1
% y la longitud de la rampa en microsegundos.

if length(points) ~= len
    fprintf("ERROR: numero de puntos del archivo incorrecto\n");
    fclose(f);
    return
end

fprintf("newramp %d\n", len);
fprintf(serial, "newramp %d\n", len);
num_chunks = 20;
for j=0:len/num_chunks-1
    chop = points(j*num_chunks+1:j*num_chunks+num_chunks);
    chop = num2str(chop);
    fprintf("addpoints %d %s\n",[num_chunks chop]);
    fprintf(serial, "addpoints %d %s\n",[num_chunks chop]);
    pause(0.05)
end
rem = len - num_chunks*floor(len/num_chunks); %remaining points
if rem > 0
    chop = points(floor(len/num_chunks)*num_chunks+...
    1:floor(len/num_chunks)*num_chunks+rem);
    chop = num2str(chop);
    fprintf("addpoints %d %s\n",num_chunks, chop);
fprintf(serial, "addpoints %d %s\n",num_chunks, chop);
    pause(0.05)
end
fprintf("setramp 1\n");
fprintf(serial, "setramp 1\n");
end
```

A continuación se puede ver un ejemplo que lee del puerto serie y escribe a un archivo de forma continua. El archivo resultante es binario, por lo que no se puede leer con un lector de texto plano. Para la lectura de este archivo se usaría la función "fread(file, datos_a_leer, 'uint16');" siendo "file" el archivo del cual se quiere leer.

```
clc, clear, close all
delete(instrfindall)
ser = serial('COM10', 'InputBufferSize', 4e3);
f = fopen('data.txt', 'w');
fclose(ser)
```

```
fopen(ser)

while 1
    data_read = fread(ser, 2000, 'uint16');
    fwrite(f, data_read, 'uint16');
end
fclose(f)
```

Durante el desarrollo del proyecto se han utilizado algunas salidas del microcontrolador para la depuración del sistema. En el conector identificado como "J6" en la PCB, el pin 2 corresponde con una señal cuadrada (color rosa en la Figura 5) cuyo nivel alto corresponde con los instantes en los cuales se esta enviando información a través del USB y el pin 3 corresponde con una señal cuadrada que cambia de nivel cada vez que el ADC empieza a adquirir (color amarillo en la Figura 5).

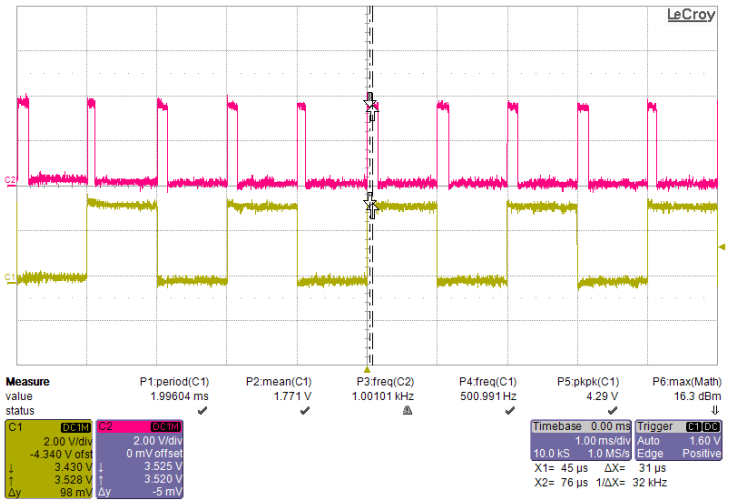

Figura 5: Señales para la depuración.