6. PLL Software: SPLL

Una de las principales ventajas que ofrece el diseño de PLLs mediante la vía software, es la flexibilidad y versatilidad de la que va acompañado un medio no físico, es decir, un programa se puede moldearlo como un LPLL, un DPLL o incluso añadirle funciones a estos últimos que sus versiones físicas no contienen. Por otro lado es necesario comentar que hoy en día los PLL hardware, presentan un amplio y barato abanico de posibilidades, por lo que entre otras cosas debe estar económicamente justificado el uso de un SPLL. Con el fin de llevar a cabo esta justificación, es recomendable llevar a cabo los siguientes pasos.

Paso 1.- Definición del algoritmo. Se lleva a cabo una presentación o bosquejo del algoritmo a implementar, con el fin de aclarar y ordenar ideas.

Paso 2.- Definición del lenguaje. Definido el algoritmo, se elije un lenguaje de programación con el fin de llevar a cabo la codificación de nuestro algoritmo. Si el programa se va a ejecutar en un microcontrolador, el lenguaje debe ser tal que exista un compilador disponible, en nuestro caso el lenguaje de programación empleado ha sido el C. Una vez compilado, se debe estimar el tiempo de ejecución. No todo compilador es capaz de generar el código ensamblado en un tiempo eficiente, este es uno de los puntos más importantes en el caso de emplear un DSP. Estos últimos, emplean técnicas pipeline, esto implica ejecutar diferentes instrucciones en una sola instrucción. En el caso de que el compilador no sea eficiente, la escritura del código debería llevarse a cabo en ensamblador, lo cual no es nada fácil.

Paso 3.- Estimación del tiempo real. Una vez estimado el tiempo de ejecución del programa, se debe calcular el tiempo real que va tardar el microcontrolador en llevar a cabo todas las instrucciones requeridas en el algoritmo.

Paso 4.- Testeo del tiempo real.

6.1 Desarrollo de un SPLL como un LPLL

A la hora de implementar un PLL, mediante vía software, lo primero que se tiene que tener en cuenta es que todas aquellas funciones que el hardware lleva a cabo de manera continua en el tiempo aquí han de ser discretizadas.

Como ya se comentó, de acuerdo con la finalidad que tiene el PLL que se ha desarrollado en este proyecto, se ha empleado como detector de fases un multiplicador de señales. La elección de éste no era otra que la naturaleza analógica de la señal de referencia, tensión de la red. Por este motivo es necesario incorporar a los bloques anteriormente comentados un convertidor analógico digital, de tal forma que la medida de la red sea digitalizada a una frecuencia de muestreo $f_s=\frac{1}{T_s}$, siendo T_s el intervalo de muestreo.

Tomando las muestras para t =0, T_s, T_s,…, nT_s. Se empleará $u_1(n)$ para referirnos a la señal de referencia muestreada en el instante t=n. Todos los bloques de nuestro sistema deben trabajar de forma síncrona con el reloj del ADC, para así tener las señales de cada uno de ellos en el mismo instante de muestreo.

En la Figura 32, se ve las entradas y salidas de cada bloque. Comentar que la señal de salida del DCO, no es $u_2(t)$ directamente. Ésta se tendrá que obtener a partir de $j_2(t)$. Si

se emplea un VCO en lugar de un oscilador controlado digitalmente, (Digital Controlled Oscillator)(DCO), su frecuencia angular instantánea sería

$$w_2(t) = w_o + K_o u_f(t)$$

A partir de esto y con la definición de fase, se puede sacar de manera continua la fase de la onda de salida del oscilador

$$j_2(t) = \int w_2(t)dt = w_o t + K_o \int u_f(t)dt$$

A diferencia de otras veces, en este caso se ha trabajado con la fase total y no con el incremento de la misma, el cual se correspondería con el segundo sumando de la última igualdad. Si se asignan los valores +1 y -1 a la onda cuadrada de salida del oscilador, se tendrá que pera el primero de los valores j_2 debe pertenecer al intervalo $(0, p)$ o $(2p, 3p)$…, Figura 31, y para el resto de los casos tomará el valor -1.

A continuación se llevará a cabo una discretización de lo anteriormente comentado.

Si se lleva a cabo un muestreo de la señal de salida del filtro $u_f(t)$, en el instante $t=nT_s$ y se asume que ésta permanece constante en el intervalo de tiempo $nT_s<t<(n+1)T_s$, la fase total del oscilador de salida variará en ese intervalo

$$\Delta j_2 = \left[w_o + K_o u_f(n)\right] T_s$$

Conocida la fase en el instante n, se podrá determinar la fase en el n+1, de la forma

$$j_2(n+1) = j_2(n) + \left[w_o + K_o u_f(n)\right] T_s$$

Así que a partir de la inicialización de la fase para el instante n=0, se podrá conocer el valor de la frecuencia en el instante siguiente. Esto es muy importante puesto que conocida la fase para t=(n+1)T, es posible obtener el valor la señal en ese instante,

$$u_2(n+1) = 1 \quad si \quad 2kp \le j_2(n+1) \le (2k+1)p$$

$$ó$$

$$u_2(n+1) = -1 \quad si \quad (2k-1)p \le j_2(n+1) \le 2kp$$

Siendo k entero.

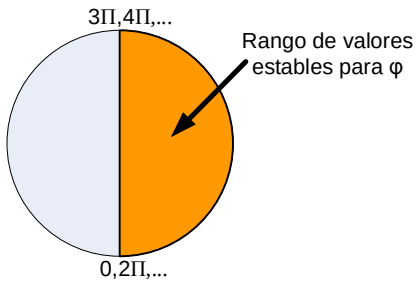

Figura 1. Rango de valores de la fase que permiten la estabilidad del sistema

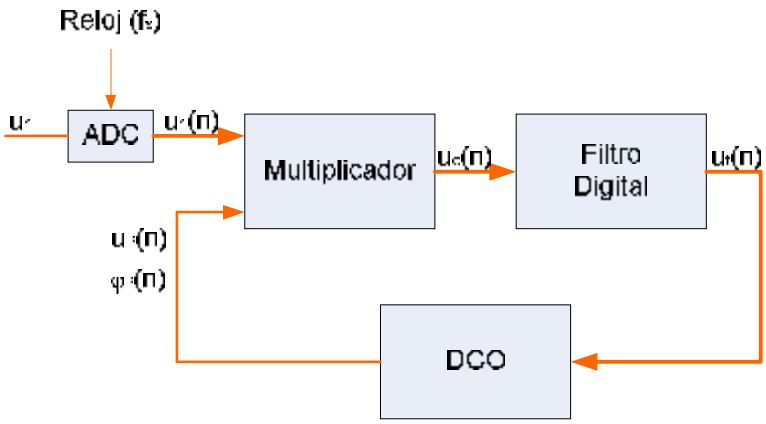

Figura 2. Diagrama de bloques de un SPLL configurado como un LPLL

En la Figura 33 se ve como evolucionan las distintas señales que intervienen en el PLL. En línea discontinua se presenta la evolución continua de las mismas mientras que los puntos representan el valor de éstas para los diferentes instantes de muestreo. Se debe indicar que tan sólo la señal $u_1(t)$ es realmente continua en el tiempo.

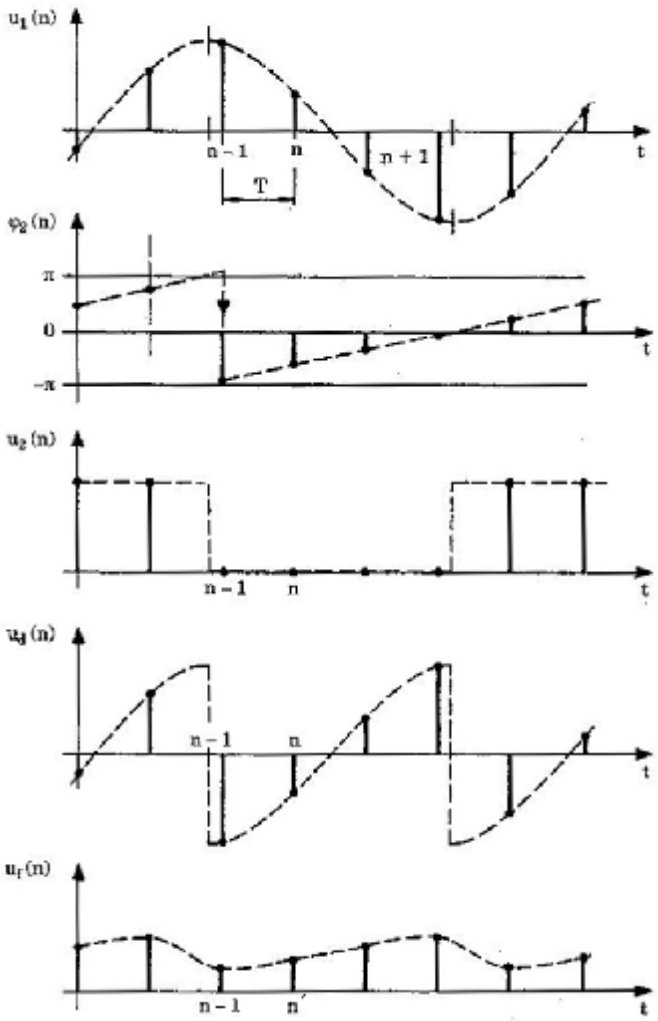

Figura 3. Dibujo de las señales calculadas por el algoritmo del SPLL

Para un instante t = nT, la salida del multiplicador $u_d(n)$ vendrá dada por

$$u_d(n) = K_d u_1(n) u_2(n)$$

Donde K_d es la ganancia del detector de fases. Conocida $u_d(n)$, en el instante T=n, un nuevo valor de la señal del filtro podrá ser evaluado en ese instante, $u_f(n)$. Conocido éste, se determinará $j_2(n+1)$, valor que se empleará para obtener el de $u_2(n+1)$ el cual es necesario para poder determinar $u_d(n+1)$…

Visto todo esto, se va a pasar a enumerar los diferentes pasos de los que va a consistir el algoritmo del SPLL:

Paso 0.- Se lleva a cabo una inicialización de las variables del algoritmo, esto es

$$u_2(n) = 0$$
$$j_2(n) = 0$$
$$u_d(n-1) = 0$$
$$u_f(n-1) = 0$$

Cada vez que se produzca la rutina de interrupción:

Paso 1.- Realizar el muestreo de $u_1(n)$ a través del ADC

Paso 2.- Operación llevada a cabo por el detector de fases: $u_d(n) = K_d u_1(n) u_2(n)$

Paso 3.- Cálculo de la señal del filtro. Un filtro analógico se caracteriza por la siguiente función de transferencia

$$F(s) = \frac{U_f(s)}{U_d(s)}$$

Siendo el numerador y el denominador las transformadas de Laplace de las señales $u_f(t)$ y $u_u(t)$. Pasemos ahora del dominio de la frecuencia al del tiempo, discretizado. Para ello se lleva a cabo la transformación de F(s) en F(z), empleando la transformación bilineal. Empleando un filtro del tipo PI, la transformación anteriormente comentada queda como

$$F(z) = \frac{b_o + b_1 z^{-1}}{1 + a_1 z^{-1}}$$

Siendo los coeficientes del filtro

$$a_1 = -1$$
$$b_o = \frac{T}{2t_1}\left[1 + \frac{1}{\tan(T/2t_2)}\right]$$
$$b_1 = \frac{T}{2t_1}\left[1 - \frac{1}{\tan(T/2t_2)}\right]$$

Donde T es el intervalo de muestreo. Con todo esto, la ecuación del filtro en el discretizada viene dada por

$$u_f(n) = -a_1 u_f(n-1) + b_o u_d(n) + b_1 u_d(n-1)$$

Paso 4.- Cálculo de la fase total del oscilador para el siguiente instante de muestreo.

$$j_2(n+1) = j_2(n) + \left[w_o + K_o u_f(n)\right]T$$

A la vista de esta ecuación se puede observar que si el algoritmo se ejecuta durante un periodo de tiempo prolongado, la fase terminará por salirse del rango permitido por los registros del DSP. Para evitar dicho overflow, se limitará j_2 en el intervalo $-p < j_2 < p$.

Paso 5.- Saturación de la fase j_2. Si cuando se muestrea dicha señal, el valor de esta excede de p , se asegura su permanencia en el rango anteriormente comentado si se resta a ésta $2p$.

Paso 6.- Obtención de $u_2(n+1)$. Dicho valor se calcula a partir de la frecuencia completa de la señal de salida del oscilador

$$j_2(n+1) \geq 0 \quad \Rightarrow u_2(n+1) = 1$$
$$j_2(n+1) < 0 \quad \Rightarrow u_2(n+1) = -1$$

Paso 7.- Se lleva a cabo una actualización de las variables empleadas. **Paso 1**.

Es importante comentar, que para evitar el fenómeno de la aliasing al implementar el filtro PLL en el PC, se debe escoger una frecuencia de muestreo al menos 4 veces mayor que la frecuenta de la señal de referencia.