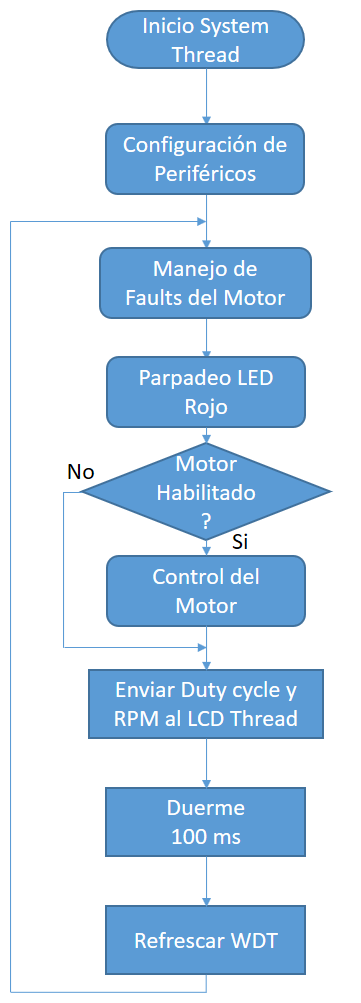
# 9. Design

En esta sección se muestran los diagramas de Flujo del diseño del software, así como un diagrama de estados del encendido y apagado del Motor de DC, también se muestra el diagrama a bloques del hardware que visualiza las conexiones entre la tarjeta de Renesas, el módulo de potencia del Motor, el motor, el potenciómetro, el LCD, los switches y los LEDs.

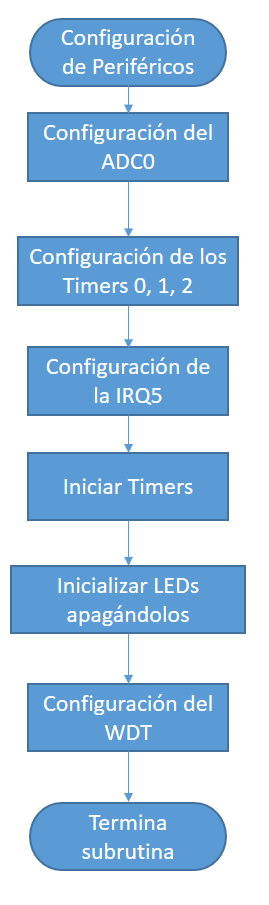
El Software del sistema Integrador está dividido en dos tareas (Threads), el System\_Thread y el LCD\_Thread el primero se encarga del sistema de control del Motor y el segundo se encarga de desplegar en el LCD los valores de RPM y ciclo de trabajo del PWM que el System\_Thread le envía.

En la Figura 9.1 se muestra el diagrama de flujo del programa principal del System\_Thread , que es el que contiene la parte de lectura del potenciómetro, lectura del sensor Hall, lectura de los switches, el control del motor, la salida del PWM y la actualización de LEDs entre otros.



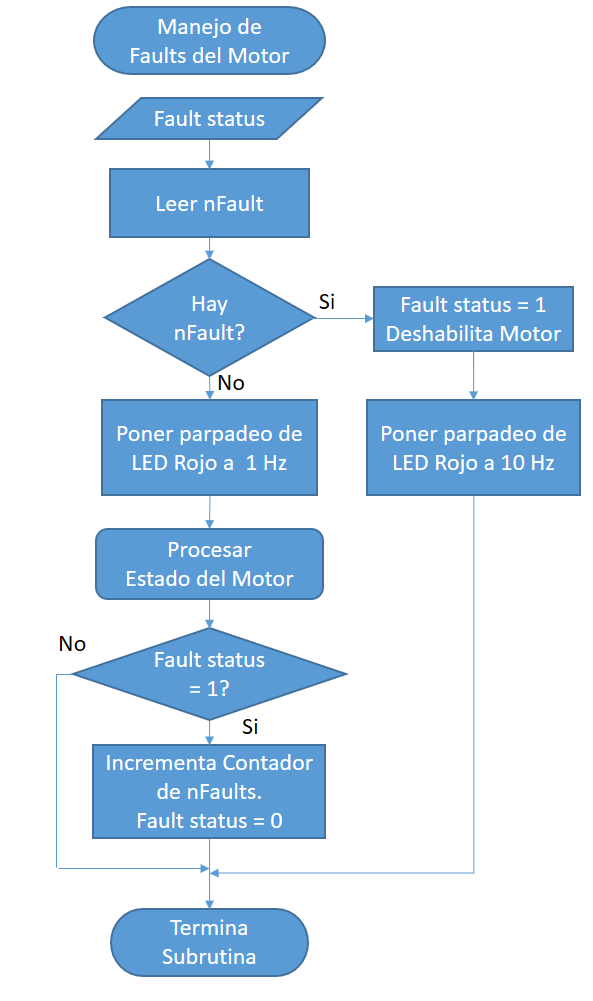
**Figura 9.1**

La Figura 9.2 muestra la configuración del ADC, de los Timers, la interrupción externa para detectar los pulsos del sensor Hall, la configuración del Watchdog timer e inicialización de los LEDs en apagados.



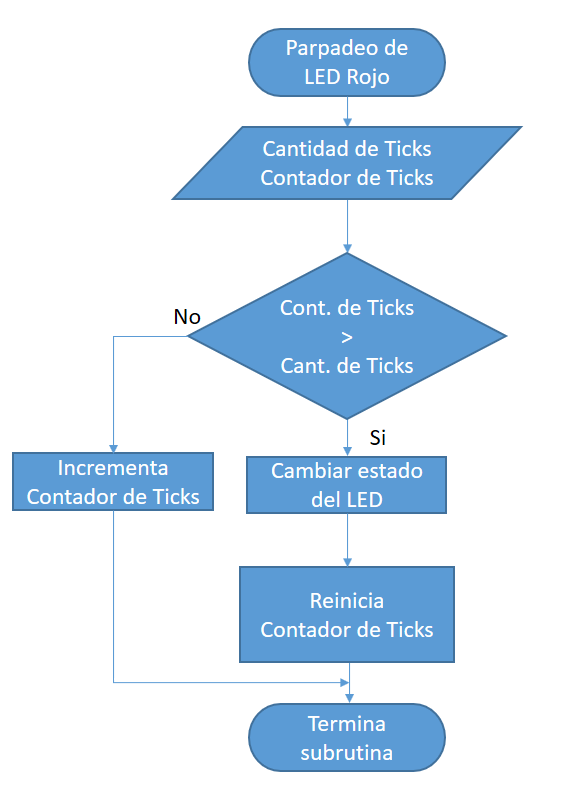
**Figura 9.2**

La Figura 9.3 muestra la subrutina del manejo del nFault del Motor, si hay un fault se apaga inmediatamente el motor y el LED rojo destella a una frecuencia de 10 Hz, cuando el motor regresa a su estado normal, el destello del LED rojo cambia su modo normal de 1 Hz y se ejecuta de nuevo la subrutina del estado del Motor la cual se encarga de monitorear los switches para encender o apagar el motor.



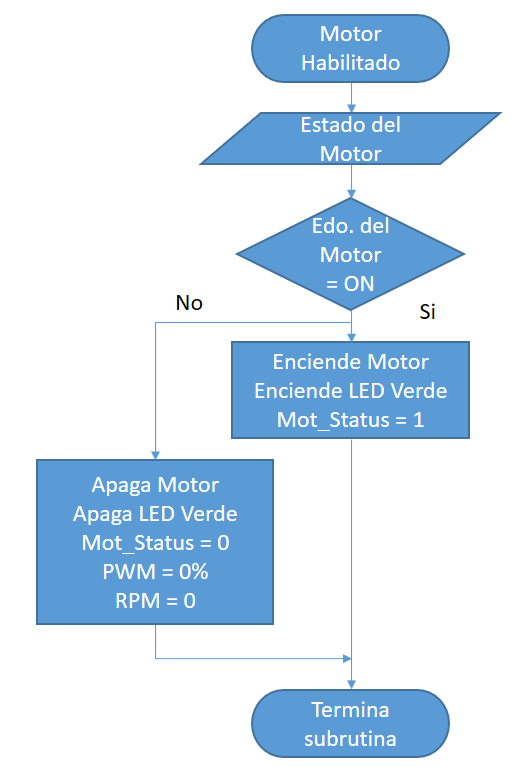
**Figura 9.3**

La figura 9.4 muestra la subrutina para el destello o parpadeo del LED Rojo, que está en función del valor que tenga la variable global “cantidad de Ticks” la cual el LED cambiara de valor cada que el contador de Ticks sea igual a la cantidad de Ticks.



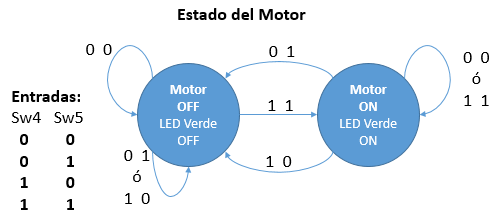
**Figura 9.4**

La figura 9.5 muestra la subrutina que enciende o paga el motor dependiendo del valor que tenga la variable global “Estado del Motor”, si esta es igual a uno, el motor se enciende, así como el LED verde, en caso de que el valor sea igual a cero tanto el motor como el LED verde se apagan y los valores que se envían al LCD también son iguales a cero.



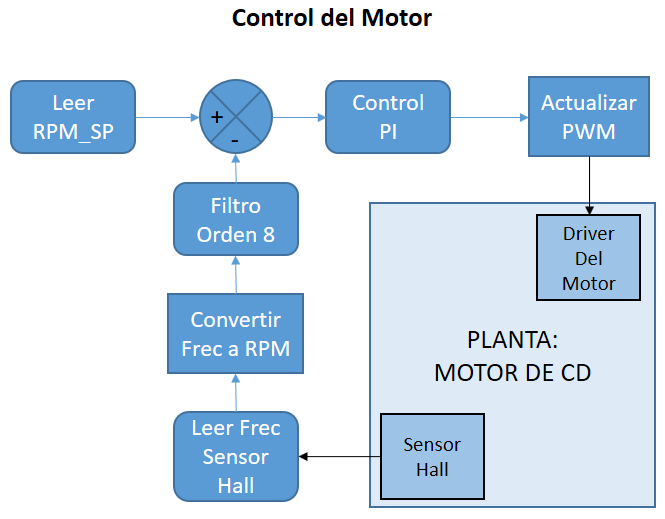
**Figura 9.5**

La figura 9.6 muestra un diagrama de estados que de acuerdo a los valores que tengan las entradas de los switches Sw4 y Sw5 será el estado del Motor: Encendido a apagado.



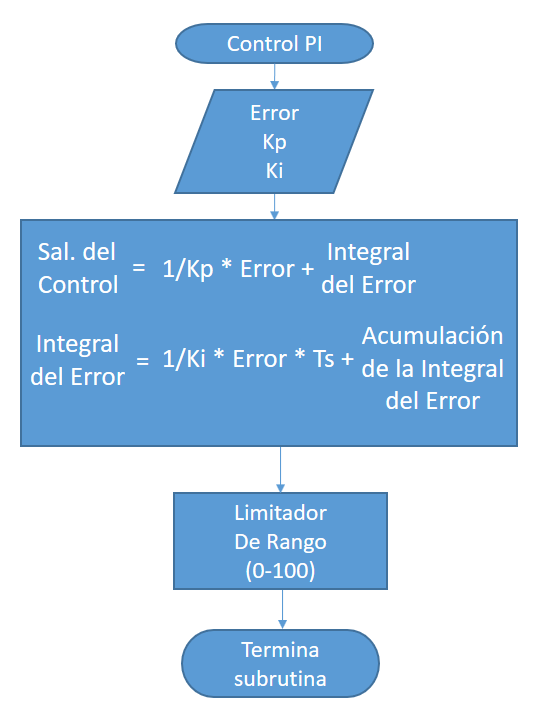
**Figura 9.6**

La figura 9.7 muestra el control del Motor el cual tiene como entradas el valor de setpoint que viene del potenciómetro leído por el ADC y del valor obtenido por el sensor Hall, el control seleccionado fue un control PI, la parte derivativa de un control PID no se está usando ya que este no se recomienda cuando hay variaciones o ruido en la señal que proviene de la planta ya que la parte derivativa amplifica estas variaciones. La salida del control PI va a un PWM ya que la velocidad será directamente proporcional al porcentaje de trabajo de la señal cuadrada que estará entre el rango de 0 a 100%, y la frecuencia del PWM (canal 2) se mantiene constante a 1 KHz.



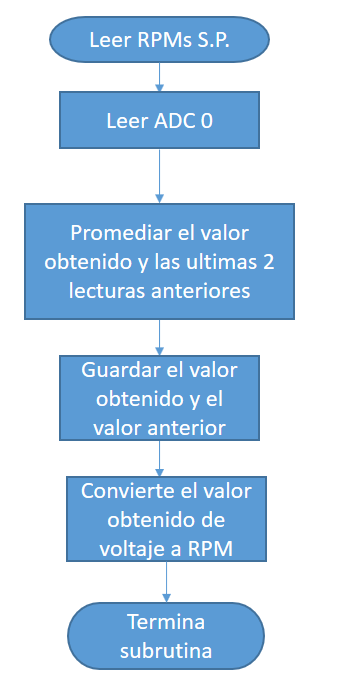
**Figura 9.7**

Figura 9.8 muestra el algoritmo del PI, el control PID es de los más comúnmente usados ya que para este tipo de controles no es necesario conocer el modelo de la planta, simplemente se hace un ajuste o sintonización de las ganancias aplicando algún método por ejemplo el de Ziegler y Nichols, cuyo método se explicará más adelante.



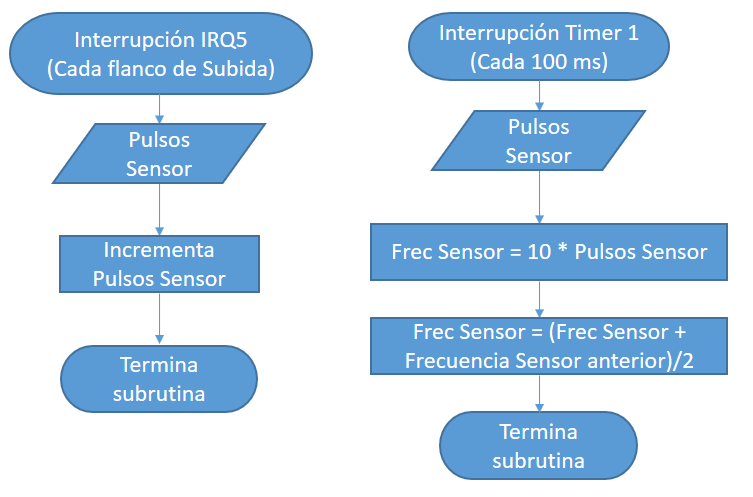
**Figura 9.8**

La figura 9.9 muestra la subrutina para leer el valor del set point en RPMs, este se realiza a través de un potenciómetro conectado al ADC canal 0, como se observa en la figura este obtiene el promedio de las 3 últimas lecturas obtenidas del potenciómetro para disminuir las variaciones, y el valor resultante es convertido a RPM.



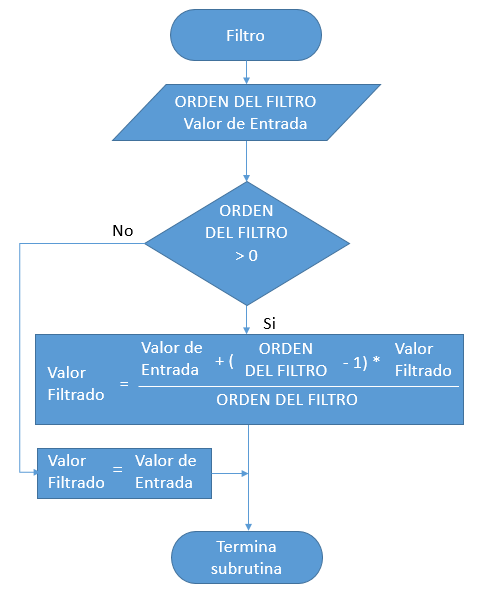
**Figura 9.9**

La figura 9.10 muestra las dos interrupciones que son utilizadas para leer el valor del sensor en RPM, la interrupción externa IRQ5 está configurada para detectar flancos de subida de la señal cuadrada proveniente del sensor, cada que ocurre la variable Pulsos Sensor es incrementada en uno, y la interrupción del timer 1 se configuró para que cada 100 ms haya un servicio de interrupción en donde se obtendrán la cantidad de pulsos leidos en este periodo, luego este valor es convertido a frecuencia y finalmente se obtiene el promedio de los dos últimos valores de frecuencia obtenidos.



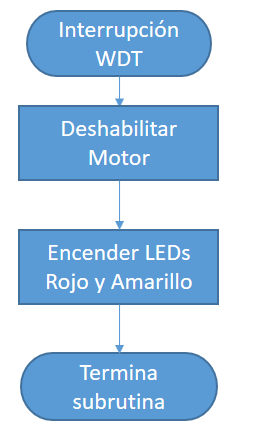
**Figura 9.10**

La figura 9.11 muestra la implementación de un filtro de orden 8, ya que en las pruebas obtenidas de la lectura del sensor se observaron variaciones aún con el promedio de las dos últimas muestras de la frecuencia.



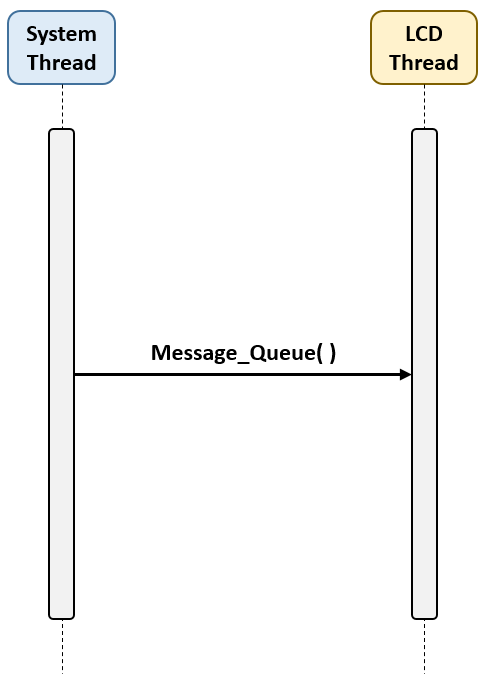
**Figura 9.11**

Finalmente, en la figura 9.12 se muestra la interrupción del watchdog timer, la cual cuando no se refresca el WDT debido a un estancamiento o bloqueo en el código, esta será llamada y apagará inmediatamente el motor y encenderá los LED Rojo y Amarillo.

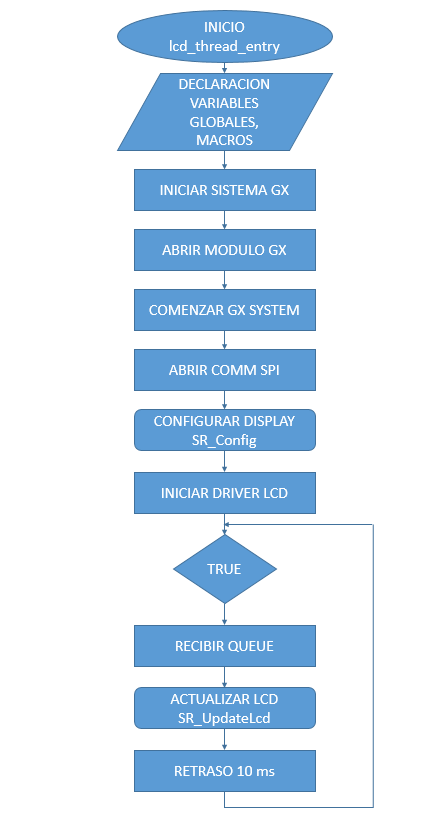
****

**Figura 9.12**

En la figura 9.13 se muestra la interrelación entre los dos Threads del Proyecto Integrador, como se observó en los diagramas de flujo (Figuras 9.1 – 9.12), el system\_Thread se encarga de la parte del control del Motor y como se verá a continuación el LCD\_Thread se encarga de desplegar los valores de RPM y ciclo de trabajo del PWM en el LCD, la comunicación entre ellos se realiza a través de las APIs tx\_queue\_send y tx\_queue\_receive que proporciona el sistema Operativo ThreadX.

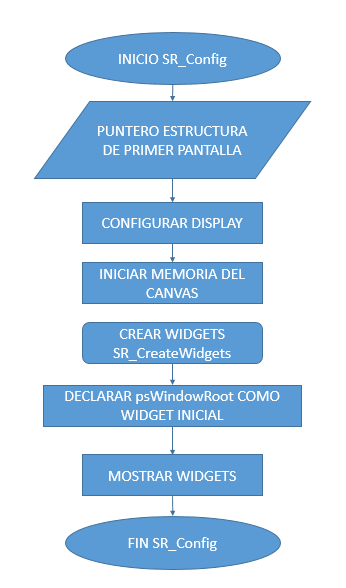


**Figura 9.13**



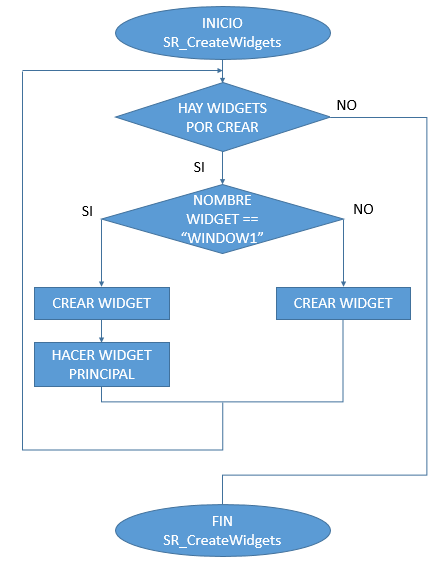
**Figura 9.14**

La figura 9.14 muestra la configuración inicial (SETUP) del thread de la pantalla LCD, en el cual se inicializan distintos módulos encargados de la comunicación SPI, la configuración del LCD, el driver del LCD, el módulo GX. También cuenta con un bucle infinito en el que se recibe la información proporcionada por el thread del sistema, los datos obtenidos se envían a los widgets del LCD en la subrutina SR\_Update\_LCD.



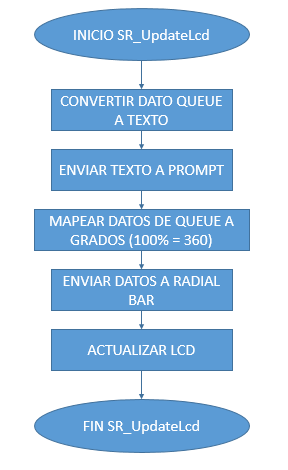
**Figura 9.15**

La figura 9.15 muestra una subrutina adicional para la configuración de los widgets que se mostraran en la pantalla. En esta subrutina se configuran e inicializan el canvas de la pantalla donde se dibujaran los widgets y se llaman otras subrutinas para generar los widgets.



**Figura 9.16**

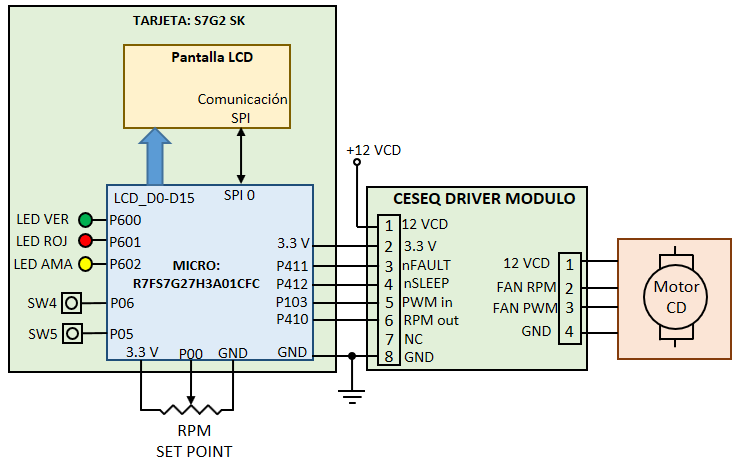
La figura 9.16 muestra la subrutina mandada a llamar en el diagrama anterior de configuración de widgets, en ésta se crean los widgets a desplegarse en la interfaz de usuario. Un ciclo evalúa los widgets a ser creados basándose en un arreglo que contiene los nombres a ser usados y cuando el arreglo es recorrido por completo dejan de generarse los widgets. Al encontrar el widget que por nombre tenga “window1” lo convierte en el widget principal, pasando su dirección de memoria al puntero psWindowRoot.



**Figura 9.17**

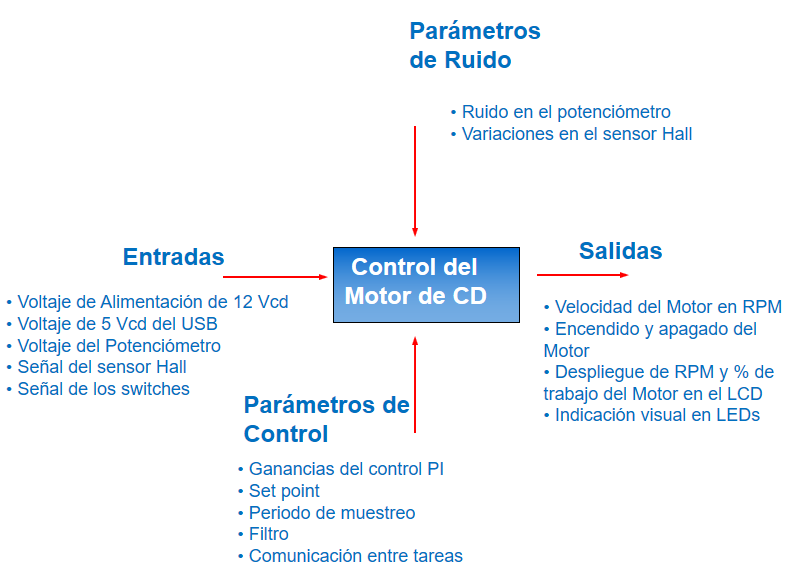
La figura 9.17 muestra el diagrama de flujo del código de actualización de datos de la pantalla LCD, es una subrutina llamada por la rutina principal del thread, la cual se ejecuta en un ciclo infinito, para constantemente actualizar los datos desplegados en la pantalla provenientes del thread principal del control.

Por el lado del hardware en la figura 9.18 muestra un diagrama de bloques de las conexiones entre la tarjeta de Renesas, el módulo de potencia, el Motor, el potenciómetro, el LCD, los switches y los LEDs.



**Figura 9.18**

Adicionalmente en la figura 9.19 muestra el diagrama de control donde se define las entradas, las salidas, el ruido y las señales de control.



**Figura 9.19**