UNIVERSIDAD NACIONAL DE MAR DEL PLATA

Diseño e implementación de un Spin Coater

Trabajo final de la carrera de Ingeniería Electrónica

Mesa, Juan María

30/07/2017

Contenido

1. Introducción

1.1 Spin Coater

El proceso de Spin Coating es utilizado para la aplicación de una película delgada de un fluido o resina sobre un sustrato. Un proceso típico involucra la deposición de una pequeña cantidad de fluido sobre el centro de la superficie del substrato, para luego ser sometido a una alta velocidad angular. Producto de la aceleración centrifuga causada, se provoca el esparcimiento del fluido dejando una película delgada sobre el sustrato. El grosor final de la película y otras propiedades dependerán de dos factores, la naturaleza de la resina y los parámetros elegidos para el proceso de spin, tales como la velocidad y aceleración angular.

Uno de los requerimientos más importante en el proceso, es la posibilidad de repetir los resultados con precisión. Variaciones sutiles en los parámetros seleccionados pueden desencadenar en variaciones drásticas en las características del producto final.

Se pretende desarrollar un equipo que le permita al usuario programar etapas o pasos del proceso que involucren distintos perfiles de aceleración y velocidad con tiempos de duración determinados. La velocidad angular juega un rol fundamental en el resultado final, definiendo el grado de la fuerza centrífuga sobre el fluido, y en particular la etapa de máxima velocidad determinara el grosor de la película. Variaciones del orden de ±50 RPM de la velocidad en la etapa de máxima velocidad pueden causar una cambio del 10% en el grosor resultante del recubrimiento. Además la programación dispondrá de una resolución de 10 RPM. Mientras que las etapas de velocidad constante proveen una fuerza radial, la aceleración sobre el sustrato aplica una fuerza tangencial sobre la resina, siendo esto de ayuda en los casos donde se requiere dispersar el fluido sobre sustratos con una superficie con un patrón o relieve proveniente de un proceso previo.

1.2 Equipo propuesto

Se propone desarrollar íntegramente (tanto parte mecánica como electrónica) un Spin Coater de bajo costo.

Con respecto a la performance del equipo, se pretende alcanzar velocidades superiores a 8000 RPM con un error no mayor a 5 RPM y etapas de aceleración constante programables con una resolución de 10 RPM/seg.

A efectos de lograr la performance anteriormente detallada, se diseñara un sistema de control con un esquema a lazo cerrado, con realimentación de velocidad, a fines de garantizar la robustez del equipo con respecto a los parámetros más críticos del proceso de Spin Coating (aceleración y velocidad). Con respecto a la interfaz de usuario, se presentara un panel con un Display y los controles necesarios.

1.3 Objetivos de la tesis

El objetivo de esta tesis es proveer de un Spin Coater al laboratorio de materiales de la Universidad Nacional de Mar del Plata.

Se pretende que a lo largo de todo el proyecto se puedan integrar los conocimientos adquiridos durante toda la carrera de ingeniería Electrónica, haciendo especial énfasis en transportar los conocimientos teóricos a la práctica.

2. Desarrollo Mecánico

2.1 Sujeción de la muestra

Se estudiaron distintas alternativas para resolver el problema de sujeción de la muestra. La primera opción, fue imitar los sistemas existentes, comerciales, los cuales utilizan vacío para sujetar la muestra al Spin Coater. Dado que la pieza en cuestión, rota a una gran velocidad angular (8000 RPM), se encontró que realizar una junta rotativa, capaz de operar a estas velocidades y mantener vacío excedía el presupuesto de dicho proyecto. Este tipo de sujeción implica complejizar el proyecto agregando poleas, compresor, juntas de vacío, y demás partes móviles.

La opción que aparece como viable, y que finalmente se adoptó, fue la siguiente: La superficie donde se coloca la muestra es una superficie perfectamente plana y pulida, igualmente ocurre con la muestra, que consta de fragmentos de vidrios. Para fijar dichos fragmentos al disco se utiliza cinta adhesiva doble faz.

La única desventaja que este sistema presenta, es que el proceso de intercambiar muestras se hace lento. Pero en este caso, el equipo está dirigido a un centro de investigación, donde se hacen experimentos ocasionales. Es decir, no está dirigido a una línea de producción donde el tiempo de operación sea un factor crítico. Por lo que la solución ofrecida al problema resulta en la mejor relación costo/beneficio para el objetivo dado. (Figura 2-1)

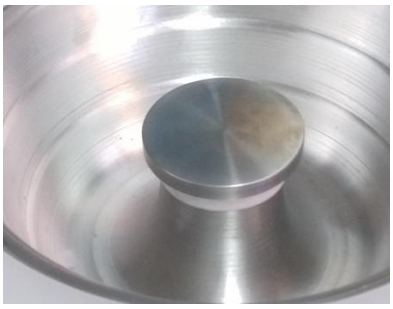

Figura 2-1

2.2 Motor

La elección del motor fue el punto de partida para el desarrollo del resto del equipo. Dadas las características de velocidad/aceleración buscadas se comenzó con la búsqueda en el mercado de un motor capaz de cumplir los requerimientos con un bajo costo.

La opción que se presento fue un Motor de corriente continua, del cual contaba con las siguientes especificaciones: (Tabla 2-1)

Vnom	52V
Potencia	450W
Velocidad Max	12000 RPM

Tabla 2-1

2.3 Rack y montaje de las partes

Se armó una estructura con caño estructural de 20X20mm, la misma se cubrió con placas de aluminio, formando el gabinete. (Figura 2-2)

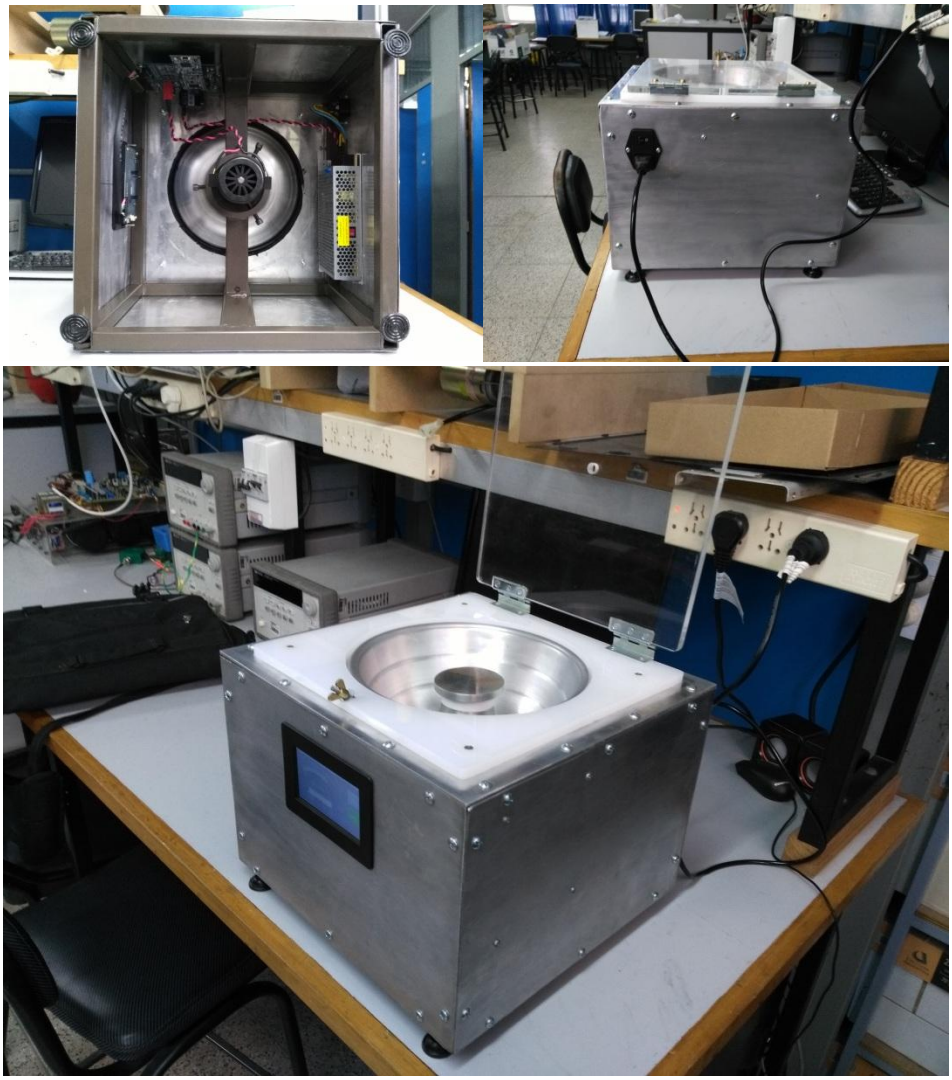

Figura 2-2

Se agregó a la estructura un soporte para fijar el motor, la cual permitió centrarlo con precisión.

Se instalaron cuatro patas de goma, con rosca, que permitieron nivelar la estructura completa, logrando de esta manera minimizar las vibraciones que pudiesen ocasionar el motor girando a más de 10.000 RPM.

Al frente del gabinete se realizó un calado donde se montó el Display TFT.

En la parte trasera se montó un conector de alimentación de 220V con interruptor integrado.

Sobre los laterales de aluminio, en la cara interior se montaron, la placa principal y la fuente de alimentación.

El recipiente que contiene los desperdicios de solución que se generan durante un proceso de deposición, se resolvió también con una pieza de aluminio, como se observa en la Figura 2-2

Dado que el proceso involucra materiales solidos girando a velocidades muy elevadas, esto conlleva un posible riesgo de accidente, que debió ser tendido en cuenta. Esto se resolvió utilizando placas de acrílico. Se armó una tapa de acrílico transparente, que permitiese al usuario monitorear el transcurso del proceso sin correr riesgos. Además se realizó una perforación, que permite la deposición de sustancias sobre el sustrato mientras el equipo está en funcionamiento.

3. Modelado de la Planta

3.1 Características de la Planta

Planta:

La planta a controlar está constituida por un motor DC con escobillas e imán permanente y un plato de aluminio solidario al eje del motor como se ilustra en la Figura 3.1.

La muestra que se coloca sobre el plato, tiene masa despreciable frente al resto del sistema, por lo que no será tenida en cuenta.

Por otro lado el motor posee un sistema de refrigeración por aire que genera un torque resistente adicional.

Figura 3-1

Motor DC:

El motor de corriente continua es básicamente un transductor de par, que convierte corriente eléctrica en par mecánico. El par desarrollado en el eje del motor es directamente proporcional al flujo en el campo y a la corriente de armadura.

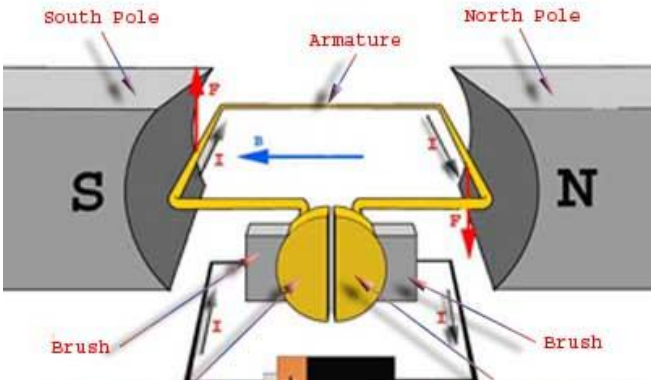

Figura 3-2

En la figura 3-2 se ilustra el funcionamiento simplificado de un motor de corriente continua con escobillas e imán permanente como el usado en este trabajo.

Un conductor, por el cual circula una corriente i_a, está inmerso en un campo magnético constante Ø, producido en este caso por los imanes permanentes. La relación entre el torque desarrollado, la corriente y el flujo magnético está dada por:

$$T_m = K_m . i_a . Ø \quad \textit{Ecuación 3-1}$$

Siendo T_m el torque del motor en N-m, i_a la corriente en amperes, Ø el flujo magnético en webers y K_m la constante de proporcionalidad.

Por otro lado, cuando el conductor se mueve por el campo magnético, se genera una diferencia de potencial entre sus extremos, que tiende a oponerse al sentido de la corriente. Este voltaje es la *fuerza contraelectromotriz* y es proporcional a la velocidad del eje. La relación entre la fuerza contraelectromotriz y la velocidad del eje es:

$$e_b = K_m . Ø . \omega_m \quad \textit{Ecuación 3-2}$$

Donde $e_b$es la fuerza contra electromotriz en volt, ω_{m} es la velocidad angular del eje en rad/seg.
Las ecuaciones 3.1-1 y 3.1-2 son la base de operación del motor de corriente continua.

Modelo matemático:

La armadura está modelada (Figura 3.1-3) como una resistencia serie R_a en serie con inductor L_a y una fuente de tensión E_a (fuerza contraelectromotriz) cuyo valor es proporcional a la velocidad de giro del eje.

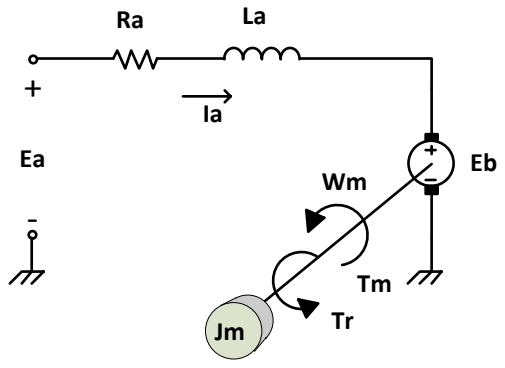

Figura 3-3

De la ecuación 3.1-1, considerando que el flujo Ø es constante dado que es generado por imanes permanentes, se obtiene el torque generado en el eje es:

$$T_m = K_t . i_a$$

Ecuación 3-3

Donde la constante de torque K_t engloba el flujo Ø y la constante de proporcionalidad K_m.
De la ecuación 3.1-2, nuevamente considerando el flujo constante, se desprende:

$$e_b = K_w . \omega_m$$

Ecuación 3-4

Donde K_w es igual a la constante de proporcionalidad multiplicada por el flujo magnético: $K_m.\emptyset$

La corriente de armadura se calcula restando la fuerza contraelectromotriz de la tensión aplicada y dividiendo por la impedancia, que esta modelada como un inductor y una resistencia serie.

$$i_{a=}\frac{e_a-e_b}{s.L_a+R_a}$$

Ecuación 3-5

Con las ecuaciones detalladas hasta aquí, ya es posible calcular cual será la corriente de armadura en función de la velocidad del eje y la tensión aplicada sobre los bornes del motor y además conocer el torque aplicado sobre el eje.

Para completar el modelo resta encontrar las ecuaciones físicas que representan las variables mecánicas del motor. Estas variables son básicamente dos: La masa del rotor y el eje, y la fuerza resistente que se opone al movimiento generado por el torque motor.

Dado que todos los componentes del motor giran con la misma velocidad angular, el momento de inercia del conjunto rotor-eje puede ser representado con una sola variable: J_m.

La fuerza que se opone al movimiento del eje, involucra más de un fenómeno físico diferente, tales como fricción de Coulomb, rozamiento del sistema de refrigeración por aire con el propio aire, el rozamiento de los rodamientos que sostiene el eje, la fuerza estática de rozamiento en el momento que el eje está sin movimiento, etc.

La ecuación integro diferencial que relaciona el torque motor y la velocidad sobre el eje es:

$$T_m - T_r = J_m.\dot{w}$$

Ecuación 3-6

Donde *w* es la velocidad angular del eje y T_r es el torque resistente que como se explico, tiene relación directa con la velocidad de giro.

Resolviendo por Laplace se obtiene:

$$w = \frac{T_m - T_r}{J_m.s}$$

Ecuación 3-7

En la Figura 3.1-4 se representa el diagrama en bloque correspondiente a la ecuación 3.1-6.

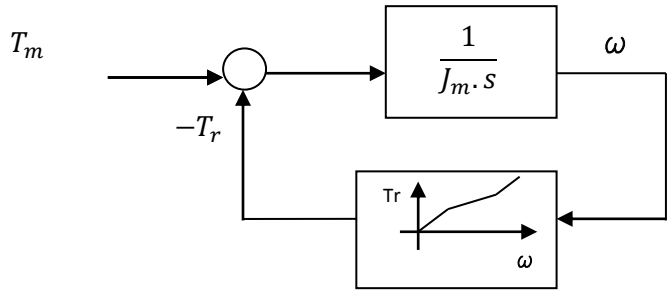

Figura 3-4

El bloque transductor entre velocidad angular y torque resistente, es completamente alineal. Dado que modelar matemáticamente todos los fenómenos que involucra reviste una dificultad considerable, como se verá en una sección posterior, es posible aproximar dicho bloque ajustando una curva a datos obtenidos empíricamente.

Otro enfoque, que es el más utilizado en la literatura, consiste en una aproximación lineal del problema definiendo un coeficiente de roce viscoso del motor B_m. De este modo la ecuación integro diferencial pasa a ser:

$$T_m - B_m.w = J_m.\dot{w}$$

Resolviendo por Laplace se obtiene:

$$w = \frac{T_m}{J_m.s+B_m}$$

El diagrama de la figura 3.1-4 se simplifica en un solo bloque (Figura 3.1-5):

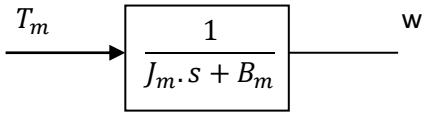

Figura 3-5

Relación entre K_w y K_t

La constante de torque K_t y velocidad K_w están estrechamente relacionadas. La potencia desarrolladla en el eje se puede expresar como:

$$P = E_b.I_a$$

O también se puede expresar la potencia mecánica como:

$$P = W_m.T_m$$

Igualando las ecuaciones 3.1-9 y 3.1-10 y desarrollando:

$$W_m.T_m = E_b.I_a$$

$$w_m . k_t . I_a = k_w . w_m . I_a$$

$$k_t = k_w$$

Por lo que de aquí en adelante estas constantes serán llamadas indistintamente como constante del motor k_m.

Finalmente, del desarrollo anterior se desprende que le modelo completo de motor es:

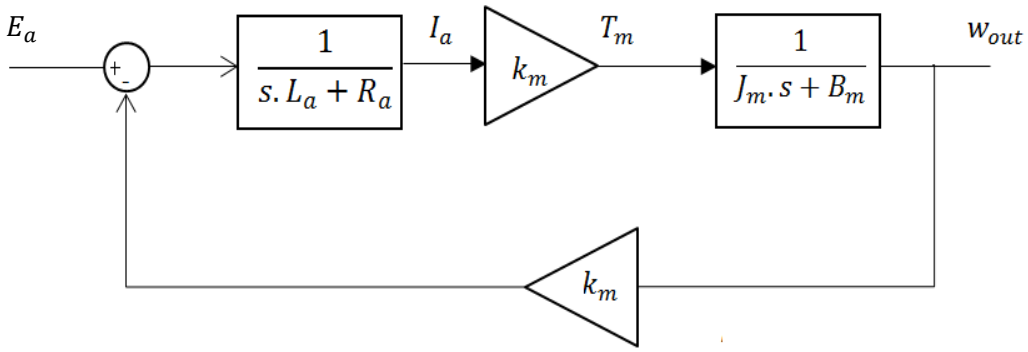

Figura 3-6

3.2 Obtención de los parámetros de la Planta

Para obtener los parámetros de la planta, una vez conocido su modelo, se recurrió a herramientas específicas de MATLAB y se montó una placa experimental que permitiese operar el motor y un banco de medición que permitiese medir sus estados (Velocidad de rotación, Corriente de armadura, Tensión en bornes)

Banco de prueba:

El banco de prueba se desarrolló con dos objetivos, primero, el de medir, caracterizar y conocer con profundidad la planta. Segundo, el de adquirir experiencia en las tecnologías a utilizar para desarrollar el hardware definitivo.

Para operar el motor se desarrolló de manera artesanal un driver del tipo puente H (Figura 3-7) que permitió operar el motor en las dos direcciones a través de PWM

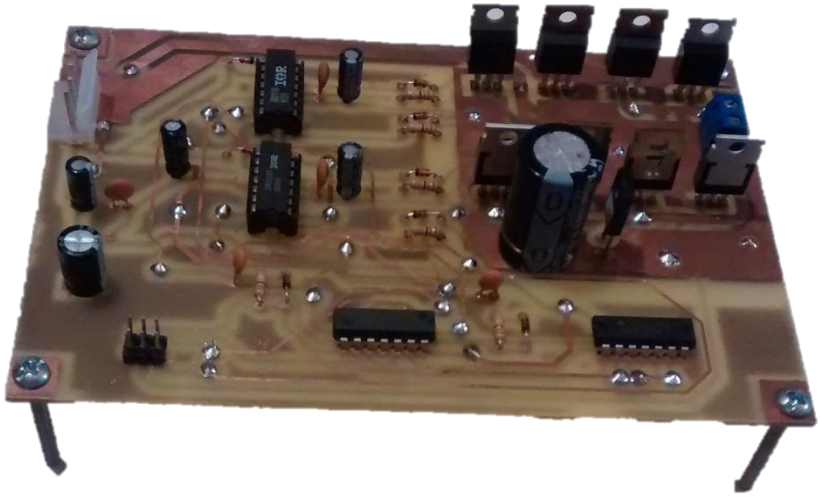

Figura 3-7

A su vez, el driver fue comandado por un microcontrolaodor montado en una placa experimental, este mismo microcontolador es el que posteriormente fue usado en el desarrollo final.

Se programó una interfaz por comandos entre dicho microcontolador y una PC, a través del software MATLAB®. De esta manera se pudo hacer pruebas del funcionamiento del motor, y de los periféricos del uControlador a ser posteriormente utilizado en el desarrollo final.

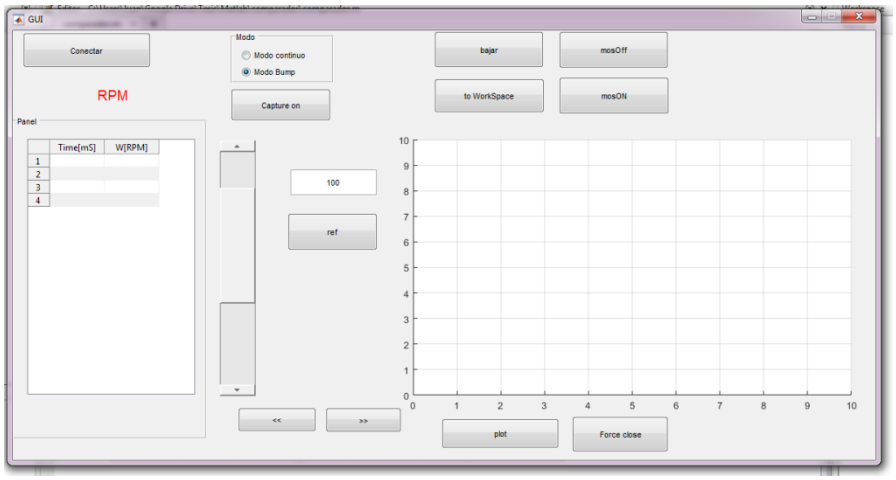

Figura 3-8

Además se levantaron las curvas características del motor, y utilizando un plugin especifico de Matlab y Simulink, se pudieron estimar todos los parámetros del modelo que caracteriza el motor.

En la Figura 3-8 se observa la interfaz gráfica GUI que se generó en MATLAB para operar el motor y colectar los datos asociados al mismo.

Par tomar la velocidad de rotación del eje se utilizó un sensor óptico.

La corriente armadura se midió utilizando un osciloscopio que permitió exportar luego los datos para procesarlos en Matlab®.

Se generó en Simulink® un modelo para el motor en base al modelo matemático que se desarrolló previamente. Como se muestra en la figura 3-9.

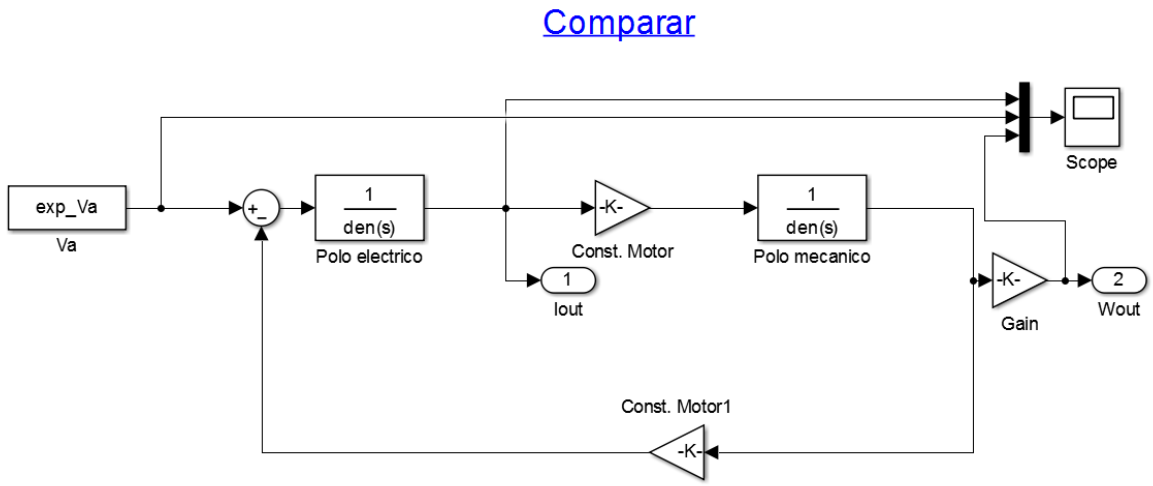

Figura 3-9

Mediciones:

Utilizando la interfaz de la figura 3-8 y el banco de prueba descripto anteriormente, se impuso una tensión Va en los bornes del motor, y se levantaron las curvas de Velocidad Wout, y Corriente de armadura Iout.

Como se observa en la figura 3-9, hay un link "Comparar" que ejecuta un script que grafica simultáneamente los datos obtenidos experimentalmente con los datos obtenidos a partir de la simulación. Se destaca que la entrada "Va" al modelo de simulación, también fue la entrada medida experimentalmente al momento de la prueba, es decir las comparaciones se hacen teniendo como entrada del sistema, exactamente la misma excitación. Las curvas obtenidas son las de la Figura 4-4.

El siguiente paso en función de determinar el modelo de la planta fue, utilizar una herramienta de Matlab y Simulink, específica para resolver estos problemas, llamada estimador de parámetros. Lo que el software hace es, a través de algoritmos numéricos de ajustes de curvas, estimar los parámetros del modelo que hacen que dicho modelo se comporte exactamente igual a los datos obtenidos experimentalmente. En este caso se los datos utilizados fueron, la tensión de excitación del motor, la corriente de armadura y la velocidad de rotación del eje. Como se observa en la figura 3-10, las curvas muestran que los parámetros que se pusieron inicialmente en el modelo realizado están lejos de acercarse al comportamiento real, medido experimentalmente.

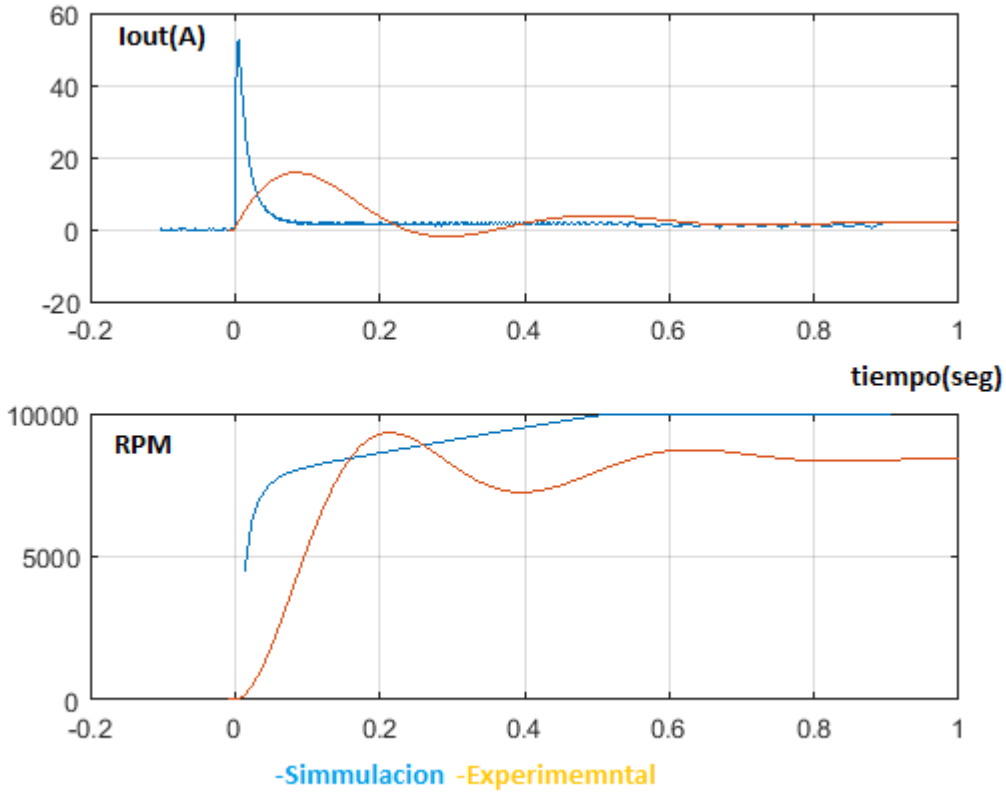

Figura 3-10

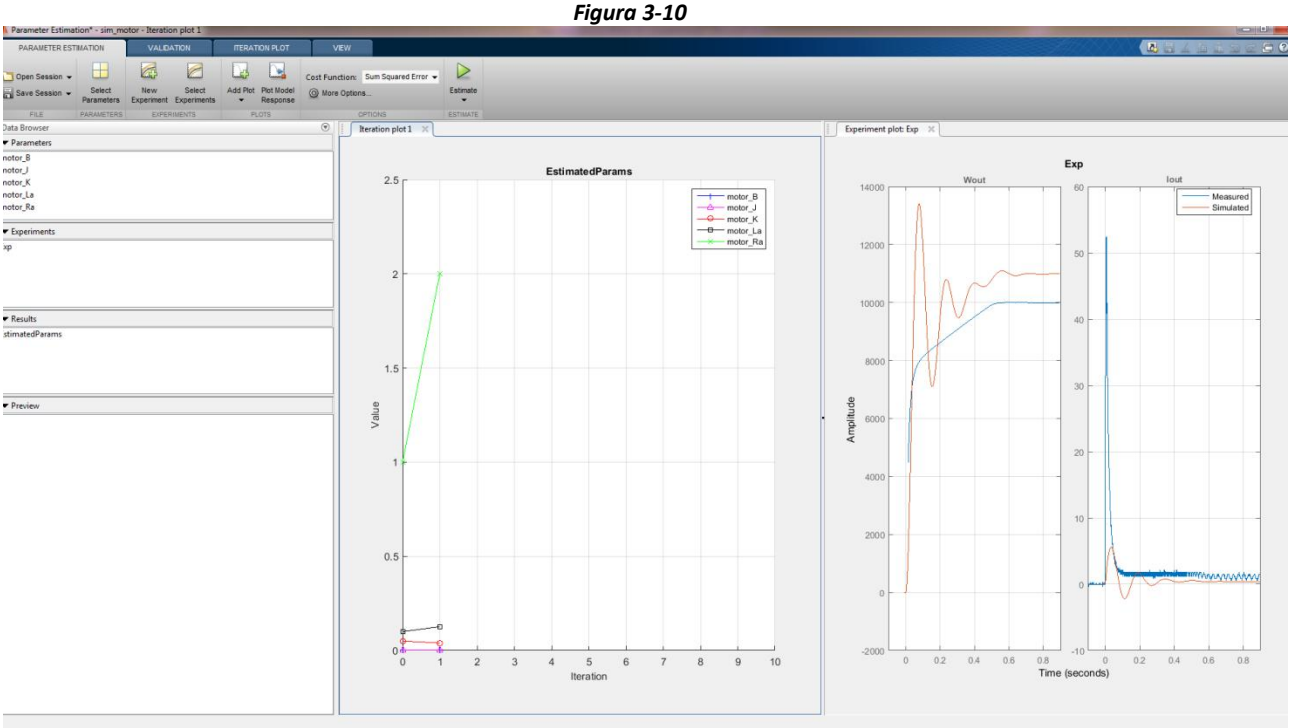

Figura 3-11

En la Figura 3-11 se muestra la interfaz que provee Matlab para estimar los parámetros del modelo armado en Simulink, una vez cargados todos los datos, se le indica al programa que inicie con la estimación. Brindándonos una gráfica con las variaciones que van sufriendo los parámetros a ser estimados y a su vez, graficando los estados del modelo que queremos aproximar a los obtenidos experimentalmente.

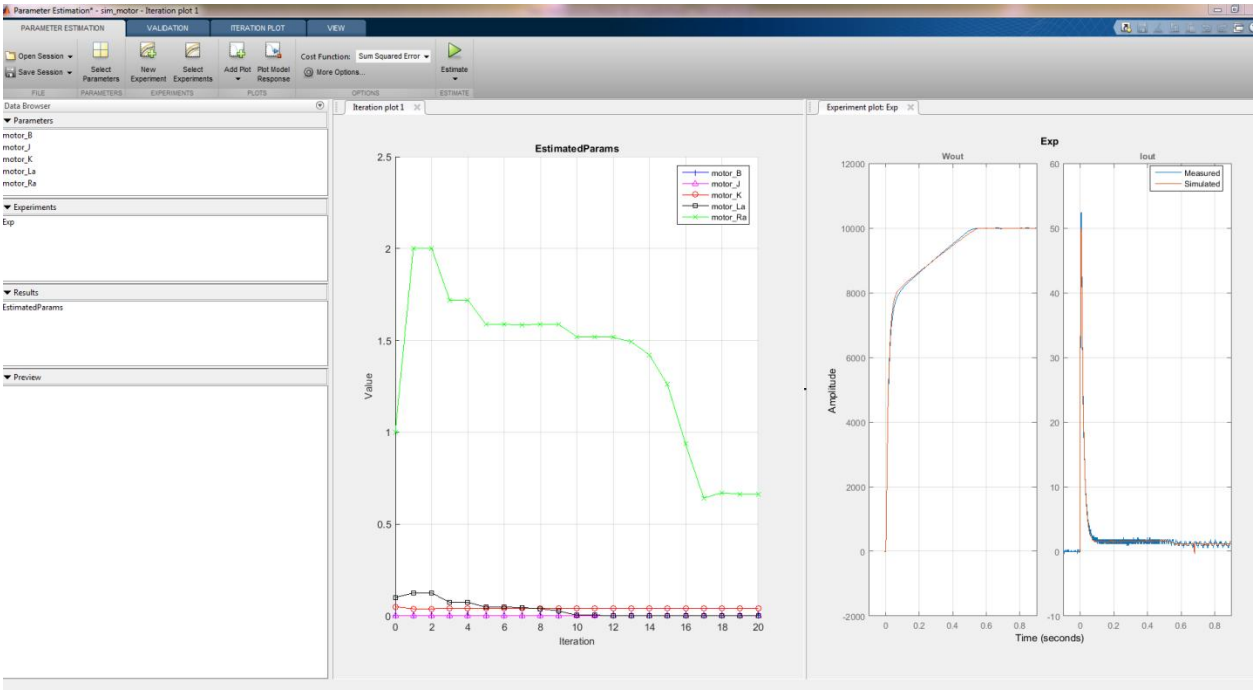

Figura 3-12

Una vez finalizado el ajuste, como se observa en la Figura 3-12, el modelo se comporta exactamente de la misma forma que la planta real. Para verificar, utilizamos el link “Comparar” previamente generado, y obtenemos los resultados de la Figura 3-13

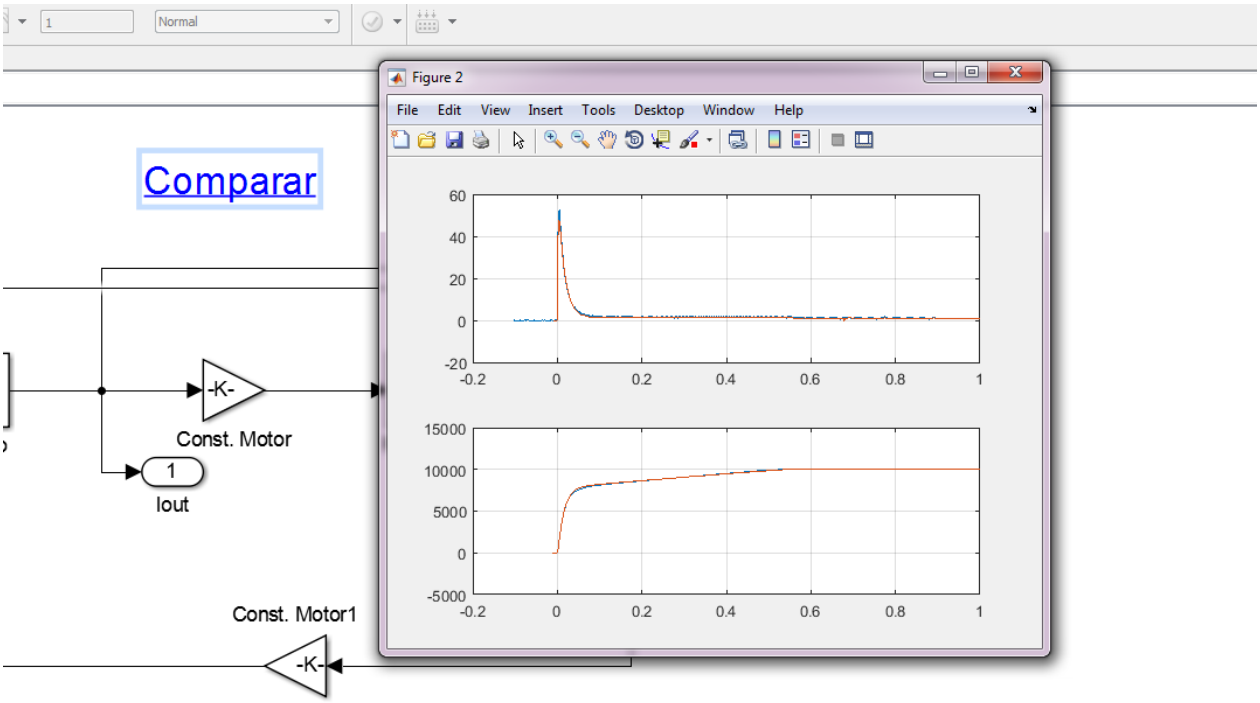
Comparar
Figure 2
File Edit View Insert Tools Desktop Window Help
60
40
20
0
-20
-0.2 0 0.2 0.4 0.6 0.8 1
15000
10000
5000
0
-5000
-0.2 0 0.2 0.4 0.6 0.8 1
-K-
Const. Motor
1
Iout
Const. Motor1
-K-

Figura 3-13

4. Desarrollo del Hardware

4.1 Introducción:

Luego de la experiencia adquirida en las tecnologías a utilizar durante el transcurso de la etapa de modelado de la planta, se procedió a desarrollar el hardware definitivo. El objetivo fue utilizar tecnologías actuales, tanto en el PCB, como en el montaje de los componentes y la interfaz HMI que se ofrecería al usuario.

4.2 Interfaz HMI:

El tipo de interfaz HMI a utilizar fue un Display Color, con tecnología Touchscreen dado que bridaría una interfaz amigable, intuitiva y facilitaría el desarrollo del proyecto desde el punto de vista mecánico (ausencia de pulsadores, luces indicadoras, etc.)

Durante la búsqueda desarrollada en el mercado en función de encontrar un dispositivo que cubriese las necesidades del proyecto, se plantearon los siguientes requisitos mínimos:

- Bajo consumo de memoria del uContolador.
- Dimensiones de la pantalla no menores a 4 Pulgadas.
- Interfaz SPI/UART.
- Disponibilidad de librerías o métodos que permitiesen una programación sencilla de la aplicación.
- Precio acorde al proyecto.

Luego de una búsqueda por los distribuidores de componentes internacionales tales como DigiKey, Mouser, etc. Se encontró un producto acorde a las necesidades, incluso con ventajas adicionales a un valor razonable.

El Display seleccionado fue un módulo para prototipos de la firma FTDI, modelo **VM800B**. (*Figura 4-1).*

Figura 4-1

Las características principales de dicho modulo son:

- Display 4.3’
- Touchscreen Resistivo
- Buzzer integrado
- Controlador FT800 de FTDI
- Interfaz SPI
- Alimentación 5V
- Soporte para MCU de 3.3V

El controlador FT800, es un procesador capaz de presentar gráficos en el Display a partir de comando enviados por SPI desde el MCU. Esto permite simplificar sustancialmente el código necesario para presentar una interfaz gráfica, permitiendo así, poder ofrecer una interfaz de buen a calidad visual con MCU’s de bajos recursos.

Todo el hardware necesario para comandar el Display y el Buzzer está embebido en el módulo, por lo que solo es necesario generar una interfaz sencilla entre la placa principal y el Display, como se detalla a continuación (Figura 4-2).

Pin No.	Name	Type	Description
1	SCLK	I	SPI Clock input, 3.3V (5V tolerant)
2	MOSI	I	Master Out Slave in, 3.3V (5V tolerant)
3	MISO	O	Master In Slave out, 3.3V
4	CS#	I	Chip select , active low, 3.3V (5V tolerant)
5	INT#	O	Interrupt output active low, 3.3V
6	PD#	I	Power down control input, active low , 3.3V (5V tolerant)
7	5V	P	5V power supply
8	3.3V	P	3.3V power supply
9	GND	P	Ground
10	GND	P	Ground

Table 3-4 – J5 Pinout

Figura 4-2

4.3 Selección del uContolador

A continuación se detallan los requisitos pretendidos para el MCU a integrar en el hardware definitivo.

- Tecnología actual.

- Bajo costo.
- Capacidad de procesamiento holgada para el proyecto.
- Herramientas de desarrollo y debugeo gratuitas.
- Fácil integración en el proyecto
- Salidas PWM, puerto SPI, ADC.

Dados los requerimientos anteriores se optó por un microcontrolaodor de la firma NXP, modelo LPC1769, el cual es un microcontolador córtex-ARM M3 que opera a una frecuencia de 100 MHZ, y cumple con todos los requisitos de periféricos detallados en los requerimientos.

Un factor importante a la hora de optar por dicho dispositivo, fue que en el mercado se consiguen a muy bajo costo, módulos de desarrollo, stick LPCXPRESSO 1769 (Figura 4-3) que incorporan el MCU, con los componentes mínimos para su funcionamiento (Cristal, Capacitores, etc.) integrados con una herramienta para programarlo y debugearlo. Esto ofrece la posibilidad de incorporar al proyecto un dispositivo de altas prestaciones, sin complejizar el posterior desarrollo y armado del PCB, ya que el LPC1769 tiene un package LQFP100.

Por último se optó por este dispositivo ya que el autor del proyecto poseía experiencia previa en el uso de tecnologías de la firma NXP.

Figura 4-3

4.4 Desarrollo de placa principal.

Se buscó desarrollar una placa que funcione como “placa base” al stick LPCXPRESSO 1769, proveyendo al mismo de los periféricos necesarios y alimentación. Para el desarrollo del PCB se utilizó el software Altium Designer. El objetivo, nuevamente fue realizar una placa que sin tener un coso elevado, cuente con la utilización de tecnologías actuales tales como compontes smd, PCB multi-layers, y además sea un desarrollo robusto en cuanto a interferencias electromagnéticas propias de circuitos que integran dispositivo s de conmutación con dispositivos digitales en el mismo circuito.

5. Diseño del esquemático

El esquemático está organizado de la siguiente manera:

Jerárquico

1. Power Supply
2. Bridge
3. CPU
4. Sensor
5. Tensiones
6. IO

Donde la hoja Jerárquico (Figura 5-1) representa las interrelaciones entre todas las demás hojas que están por debajo de la misma.

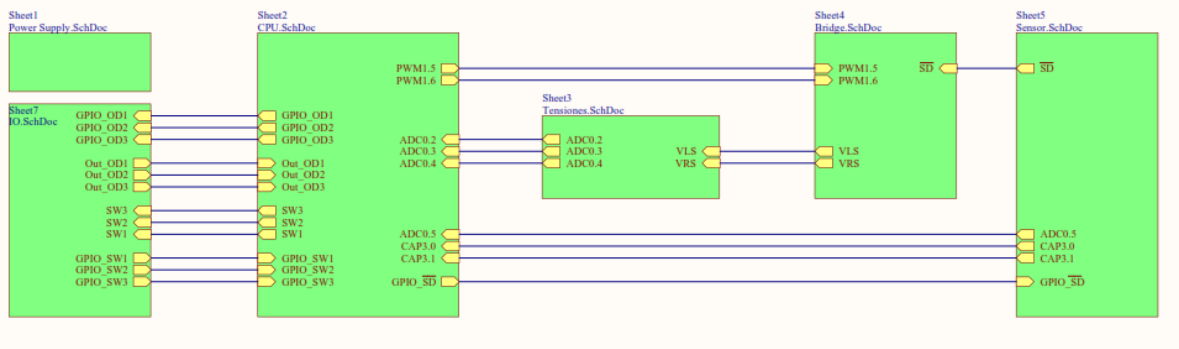

Figura 5-1

1. Fuente de Alimentación

La primer hoja dentro de jerárquico, power supply (Figura 5-2), corresponde a la fuente de alimentación interna de la placa. Se pensó un esquema donde al circuito ingresa una única tensión, y luego es convertida a todas las tensiones necesarias.

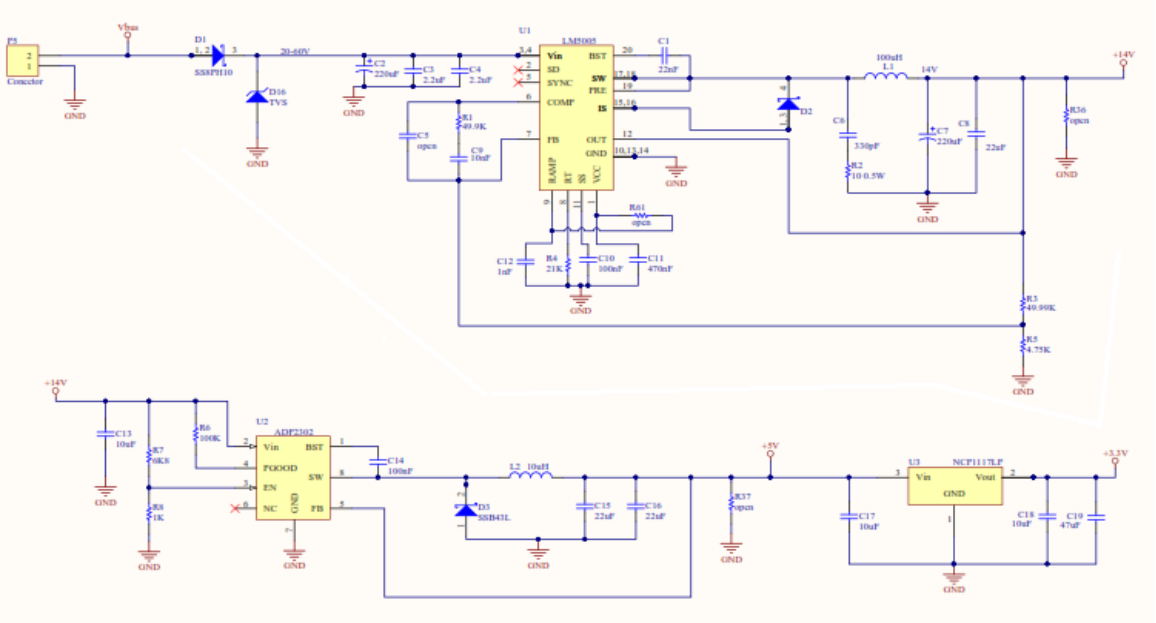

Figura 5-2

Dado que el motor tiene una tensión nominal de 52V, esta es la tensión que ingresa a la placa, Llamada en nuestro esquema Vbus. Las tensiones requeridas para nuestro sistema son:

- Vbus (20V – 60V)
- +14
- +5
- +3.3

La primera etapa consta simplemente de un diodo en serie luego de Vbus y un diodo supresor de TVS (Figura 5-3) para proteger el resto del circuito de los golpes de tensión ocasionados por los arranques y frenados del motor, los cuales generan cambios indeseados en la tensión del bus.

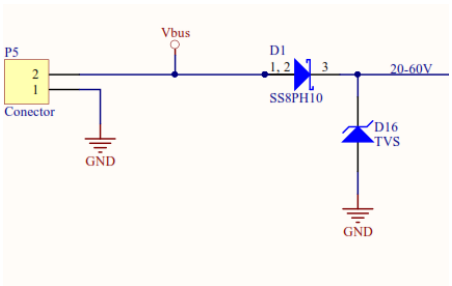

Figura 5-3

Posteriormente se realizó un esquema en cascada de fuentes para obtener +14, +5 y +3.3 respectivamente.

En la segunda etapa como se observa en la (Figura 5-4) se ubica el convertidor DC-DC de 60V a 14V. Se utilizó el circuito integrado LM5005 de la firma Texas instrumentos el cual es utiliza una topología de tipo Buck, integra dentro del circuito un conmutador de 75V y 2.5A de canal N, tiene posibilidad de configurar tensión de salida, arranque suave, protección térmica entre otras

características.

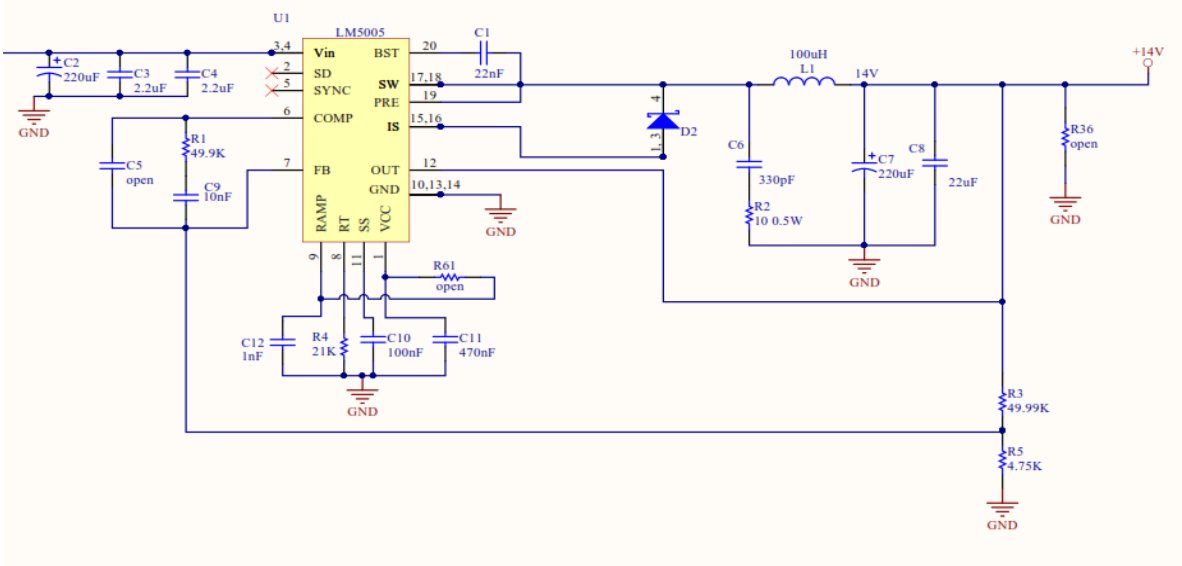

Figura 5-4

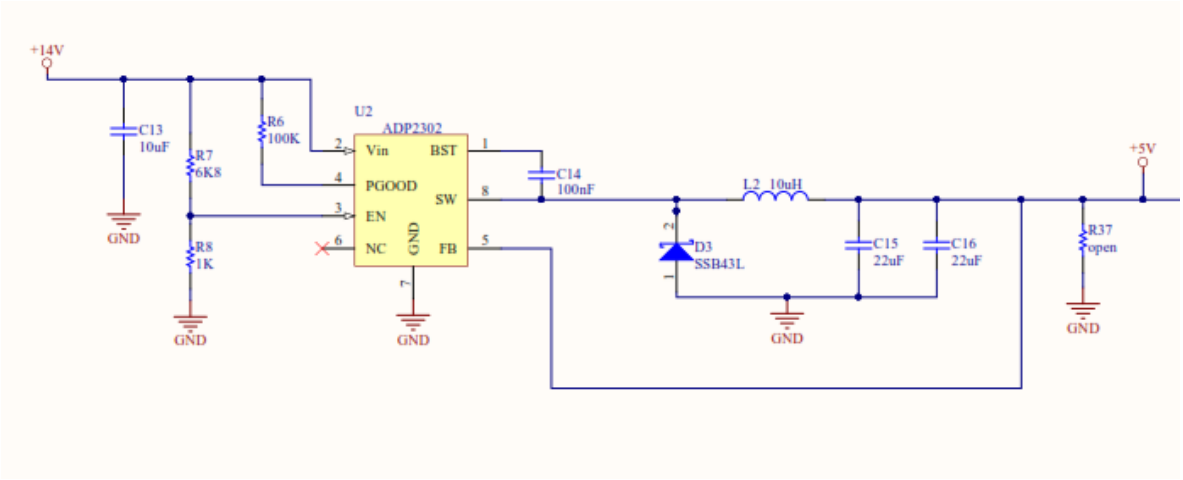

Figura 5-5

En la tercera (Figura 5-5) etapa se convirtió de los 14V previos a 5V, utilizando un convertidor DC-DC con topología Buck.

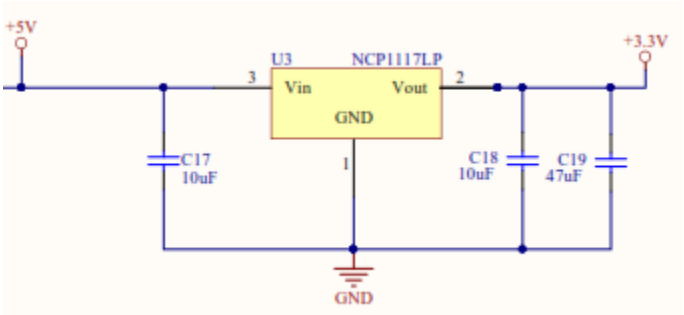

Figura 5-6

Por último, se situó la etapa de 3.3V, que se logró con un regulador lineal NCP1117. (Figura 5-6)

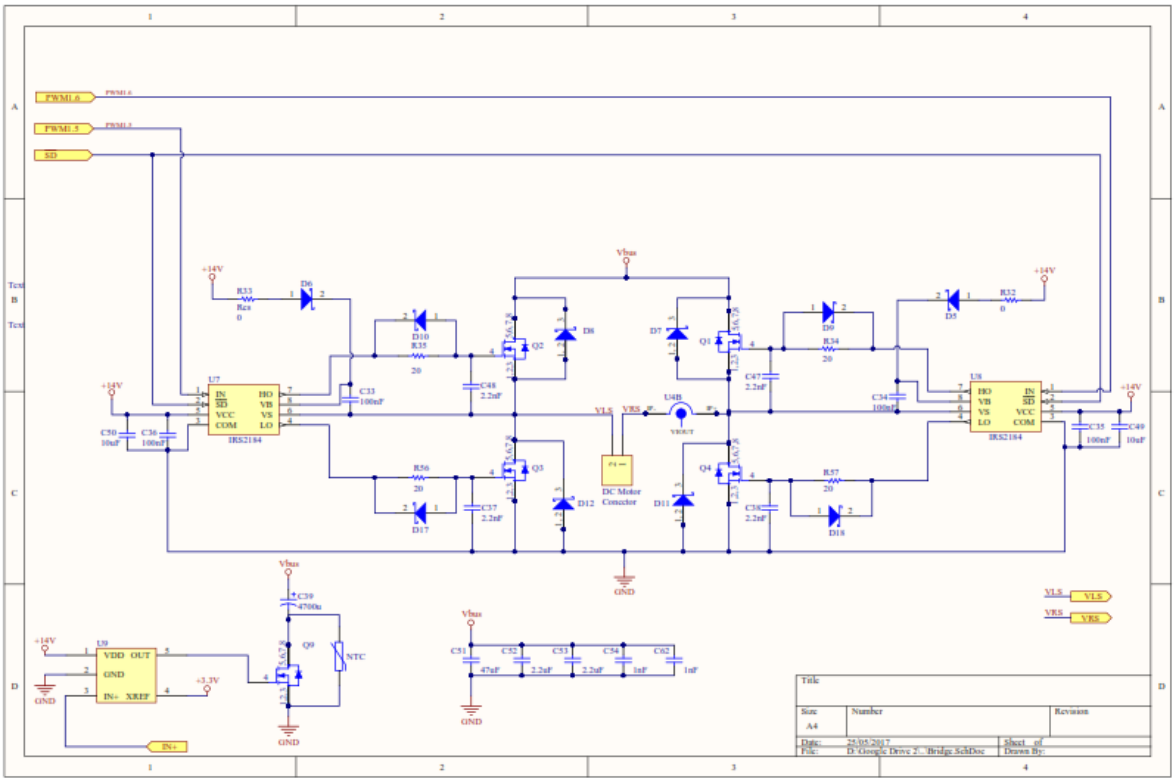

Figura 5-7

2. Driver motor DC

La siguiente hoja de esquemático (Figura 5-7) corresponde al driver que opera el actuador de la planta, el motor DC. Para lograr un control preciso, rápido y en ambos sentidos se implementó un Puente – H operado por PWM.

Las características a cumplir con el diseño fueron (Tabla 5-1):

Tensión Nominal	52V
Corriente máxima	+/-10A
PWM	20Khz
Tensión Max BUS	60V
Protección por sobre corriente	10A max

Tabla 5-1

El primer paso, a partir del cual se proyectó luego el diseño del puente, fue seleccionar los dispositivos de conmutación adecuados. En base a los requerimientos de las primeras dos filas de la Tabla 5-1, se seleccionó como dispositivo de conmutación los MOSFET de canal N DiR826ADP de la marca VISHAY®. En la Tabla 5-2 se muestran algunas de las características más importantes del dispositivo:

Drain-Source Voltage	80V max
Gate-Surce Voltage	+/-20V max
Continous Drain Current	60A max
Rds(on)	5,5mΩ (@Vgs = 10V)
Gate-source threshold Voltage	1,2V min

Tabla 5-2

El puente se diseñó de manera completamente simétrica, en la Figura 5-8 se muestra el lado derecho:

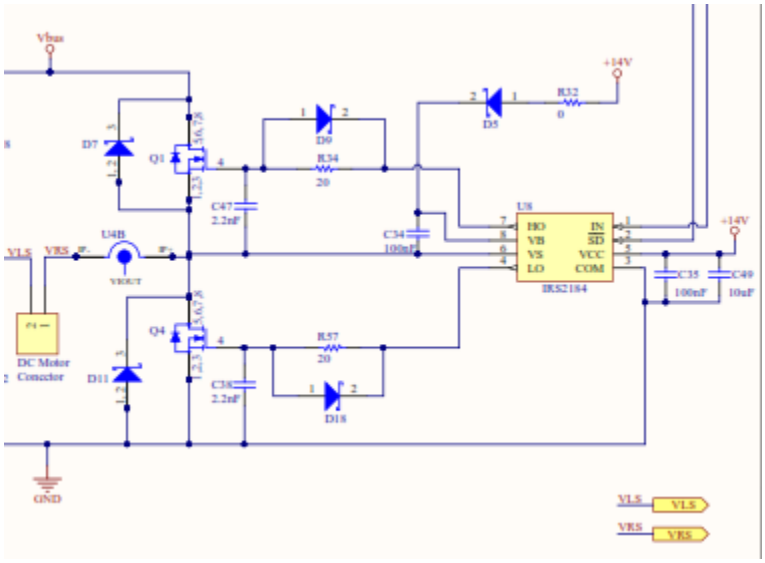

Figura 5-8

En paralelo con cada MOSFET se colocaron diodos schottky para proteger los dispositivos de conmutación en condiciones de falla, donde las corrientes inductivas puedan ser drenadas por los mismos. Se eligieron diodos schottky dado su baja corriente de recuperación inversa, ya que en esta aplicación resulta crítica para no empeorar la disipación de energía, dado que por la topología misma del circuito, dicha corriente seria absorbida directamente por los conmutadores.

Como driver para los conmutadores se utilizó el integrado IRS2184 de la firma International Rectifier®. Este integrado posee la capacidad de manejar de manera sincrónica medio puente, con señales lógicas 3.3v - 5v compatibles. Para lograr la tensión gate-surce en el conmutador del lado superior del circuito, el dispositivo utiliza una topología boostrap que le permite elevar la tensión entre 10 y 20V con respecto al surce del MOSFET en cuestión.

Por último se aprecia (Figura 5-9) que se colocan una serie de capacitores entre Vbus y GND. Respecto de los capacitores C51..C54,C62 , fueron colocados para luego ser ubicado estratégicamente en el circuito de modo de cerrar un lazo de alterna lo más pequeño posible para minimizar las interferencias que las conmutaciones del puente puedan causar. Por lo tanto se seleccionaron

capacitores con bajo ESR, capaces de proveer de manera instantánea la corriente alterna que consume el puente.

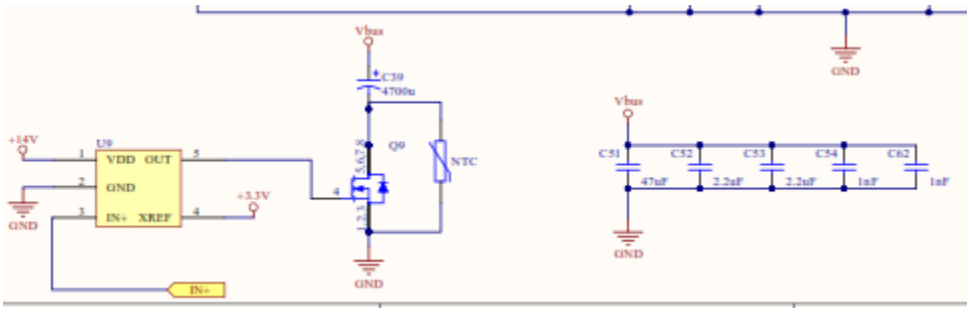

Figura 5-9

Respecto de C39, fue un capacitor electrolítico, dada su gran capacidad, pensado para sostener la tensión del BUS en condiciones de arranque y frenada del motor, dado que la fuente externa usada no era capaz de entregar toda la corriente que el circuito y el motor fueron capaces de consumir en lapsos cortos de tiempo.

Dada las dimensione del capacitor C39 fue necesario montarlo junto a un circuito limitador de corriente de arranque. Esto se logró colocando un NTC en serie con el capacitor, y un MOSFET en paralelo con el NTC conectado a un GPIO del microcontrolaodor. De modo que al encender la fuente, el capacitor se carga suavemente atreves del NTC, luego, el microcontolador habilita el MOSFET quedando el capacitor entre Vbus y GND.

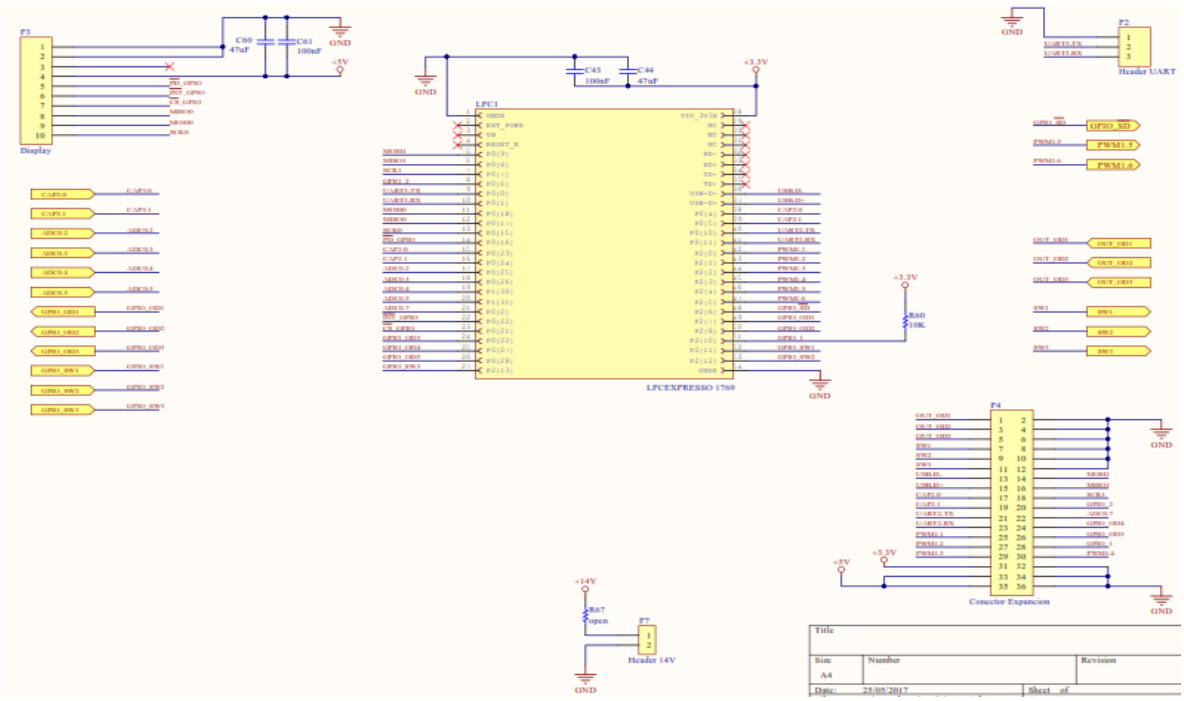

Figura 5-10

3. CPU:

Se generó un componente que represente el stick LPCXPRESSO1768, donde se mapearon todos sus pines. Luego se expresó con puertos todas las intercomunicaciones de la capa CPU con el resto de los esquemáticos (Figura 5-10)

Además, se realizaron las conexiones entre el microcontrolaodor y los periféricos tales como:

- Interfaz HMI (P3)
- UART (P2)
- Pines de expansión(P4)

Por último se observa que entre VCC y GND se conectaron dos Capacitores, que son de Bypass, para ofrecer mejor inmunidad al ruido en la placa LPCEXPRESSO1768.

4. Sensor

En la cuarta hoja de esquemático se realizaron los circuitos correspondientes a varios sensores asociados al equipo que se detallan a continuación:

Sensor de corriente de armadura del motor

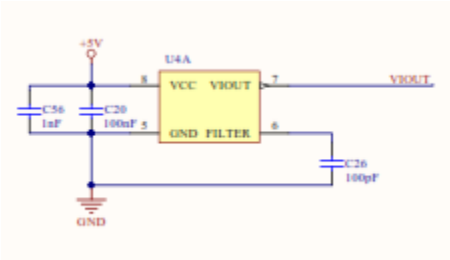

Figura 5-11

En la Figura 5-11 se observa el esquemático correspondiente al sensor de efecto hall que se utilizó para medir la corriente de armadura, el mismo fue el ASC724, de la marca Allegro®.

El esquemático de dicho componente se realizó en dos partes, siendo la parte restante la que se observa en la figura Figura 5-8 como componente U4B, representando que el componente esta serie con la salida del Puente-H.

Generador de pulsos proporcionales a la velocidad de rotación del motor

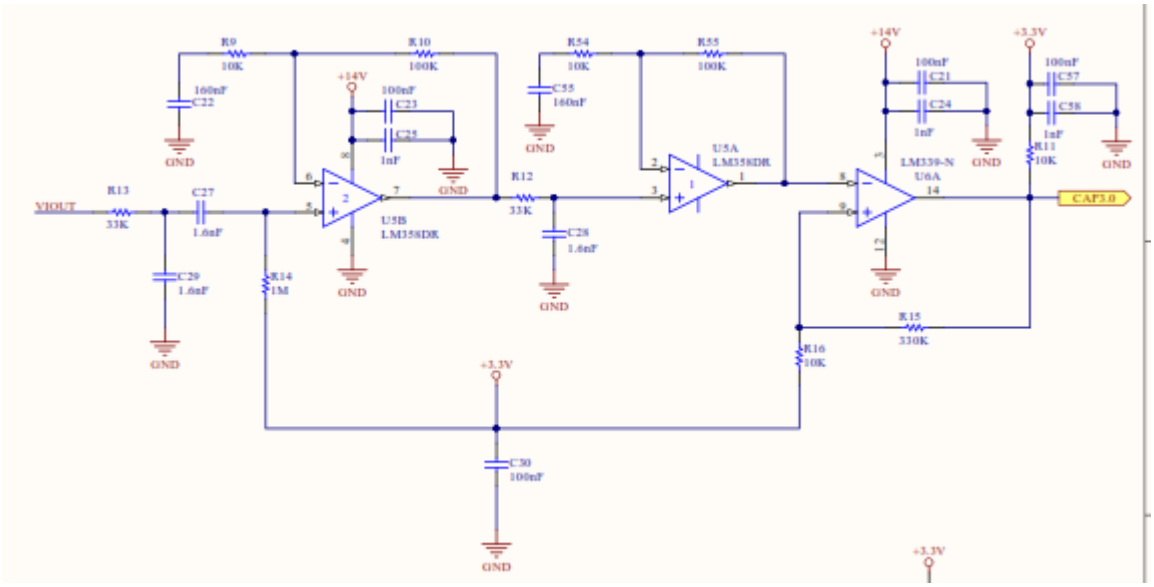

Figura 5-12

En la Figura 5-12 se observa una parte fundamental del desarrollo de este proyecto. Se trata de un sensor de velocidad realizado, en basa a la forma de onda de corriente del motor. Como se vio anteriormente, el motor utilizado fue un motor DC, dicho motor genera perturbaciones en la corriente que lo atraviesa que están directamente relacionadas a su velocidad de rotación.

Dado que al girar, este tipo de motores conmutan entre sus distintos bobinado de manera mecánica, en dichos saltos se generan perturbaciones en la corriente que al ser observados en un osciloscopio, se ve como una forma de onda senoidal montada sobre la corriente continua que está consumiendo el dispositivo.

Este fenómeno fue aprovechado para diseñar un estimador que a partir de dicha forma de onda, genere pulsos que luego atreves de una entrada de INPUT CAPTURE del microcontolador pueda estimar la velocidad de rotación del eje con una gran precisión.

Para lograrlo se realizó un circuito que fue un filtro activo pasa-banda seguida de un comparador Que genera pulso entre 0 y 3.3V, dado que es una señal apta para ser recibida por el microcontrolaodor.

Circuito de limitación de corriente de salida del puente-H.

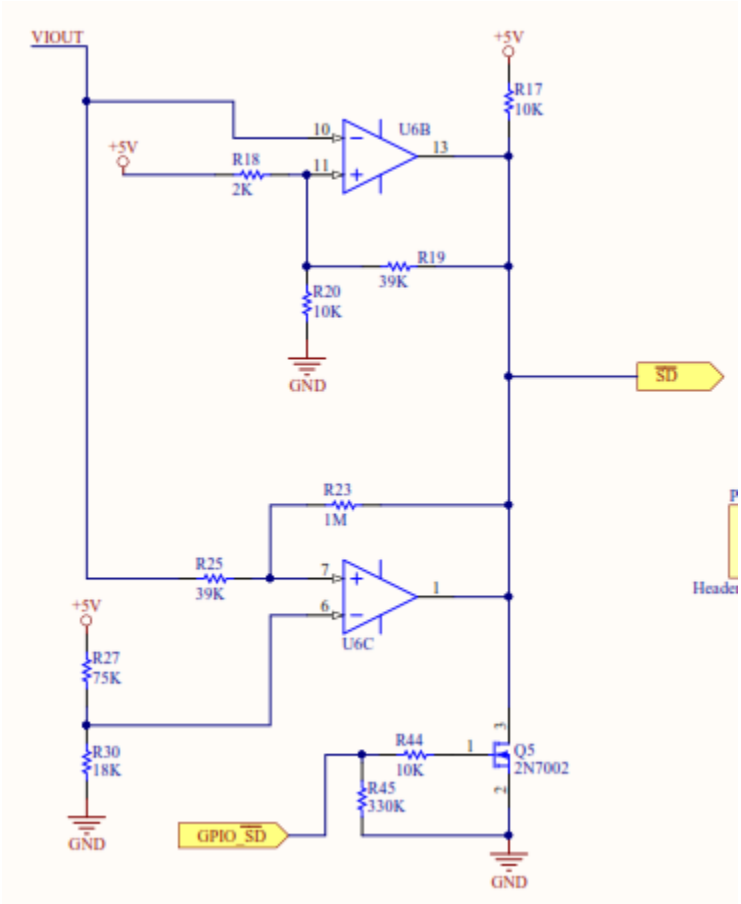

Figura 5-13

La Figura 5-13 muestra cómo se resolvió por hardware un comparador con histéresis para mantener la corriente de armadura siempre dentro de los límites seguros.

Además se conformó una “OR” con un GPIO del uC para dar la posibilidad de deshabilitar el puente por software.

Entrada con histéresis para encoder externo

Se agregó al diseño (Figura 5-14), una entrada externa, conectada a un comparador con histéresis con el objetivo de dejar el circuito preparado para un eventual caso en el que se deseara utilizar un sensor de velocidad externo en vez del estimador de corriente desarrollado previamente.

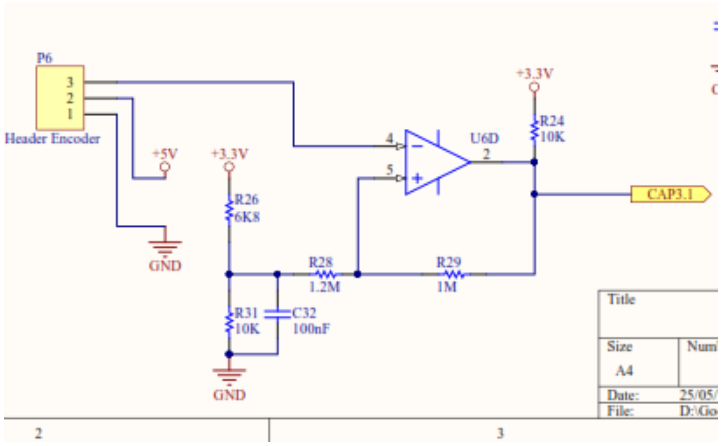

Figura 5-14

Salida de tensión proporcional a la corriente de armadura para ADC del microcontrolaodor

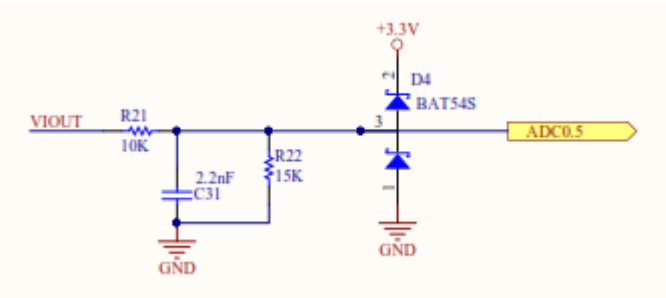

Figura 5-15

Por último, se diseñó una conexión entre la tensión proporcional a la corriente de armadura, y un conversor ADC del uC. Como se observa en la Figura 5-15 se colocó un filtro pasa-bajos, dado que nos interesa tener información de la corriente promedio, seguido de un par de diodos para proteger la entrada del uC en caso de sobretensión.

5. Tensiones

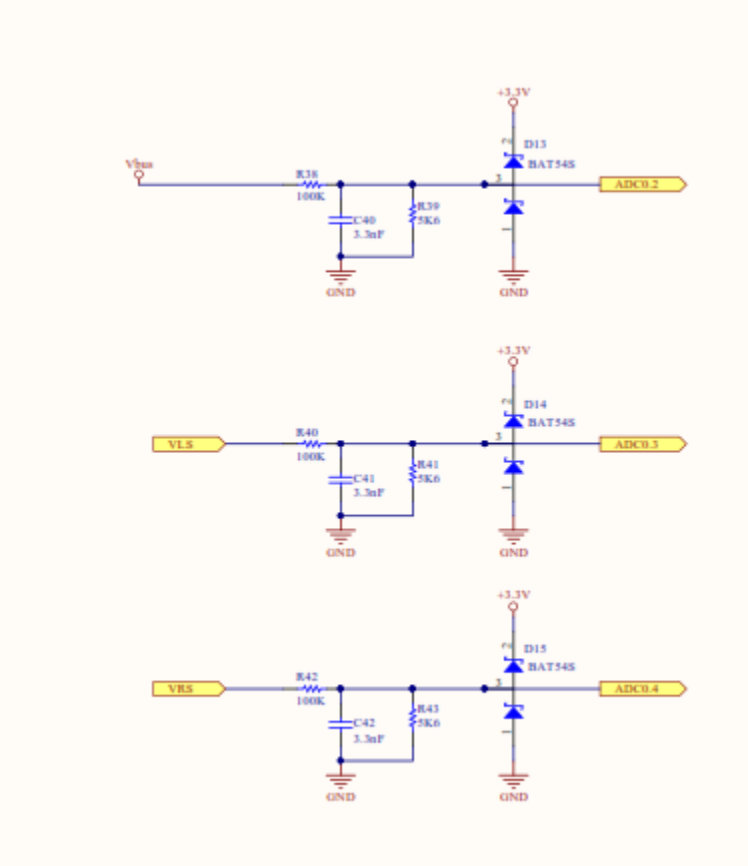

Figura 5-16

Se acondicionaron, previa entrada a los ADC del uC, las tres señales más relevantes del sistema que fueron (Figura 5-16):

- Vbus
- VLS (Tensión de un borne del motor)
- VLR (Tensión de otro borne del motor)

En los tres caso lo que se hizo es diseñar un divisor resistivo, conjunto con un filtro pasa bajo seguido de un par de diodos de protección.

5. IO

Por último se encuentra la hoja de esquemático IO(Figura 5-17). En dicha hoja se diseñaron entradas/salidas de propósito general, para conectar LED's, Switchs etc.

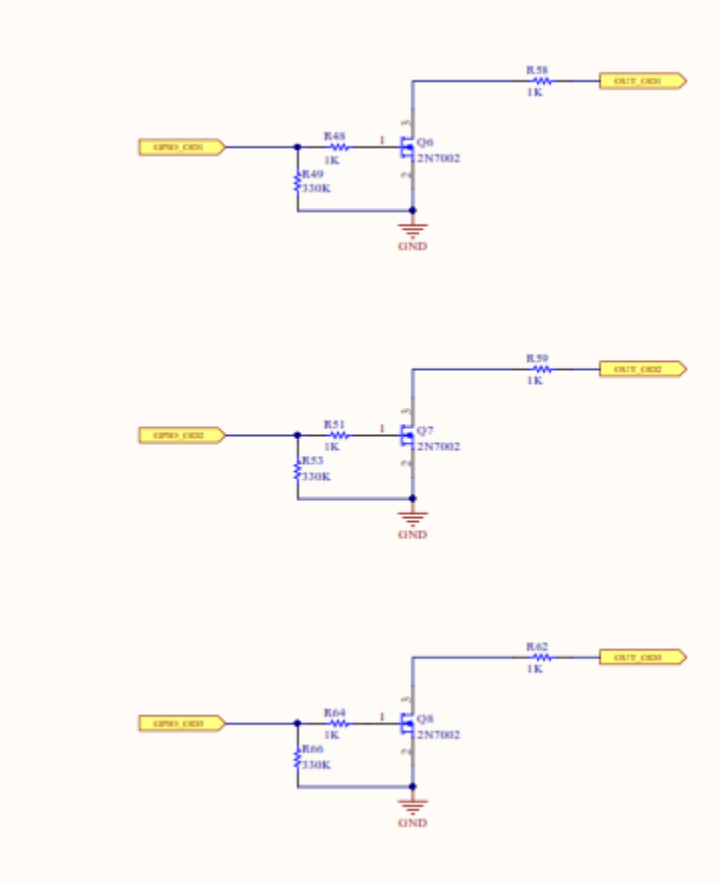

Figura 5-17

Finalmente en el proyecto no fueron utilizados dichos periféricos.

Layout y PCB

El Ruteo del PCB se desarrolló en una placa de 4 capas de 10X12 cm. Las capas se organizaron de la siguiente manera (Tabla 5-3):

Superior	Fuente alimentación/CPU
Intermedia 1	Ruteo señales
Intermedia 2	GND
Inferior	Sensores/Señales

Tabla 5-3

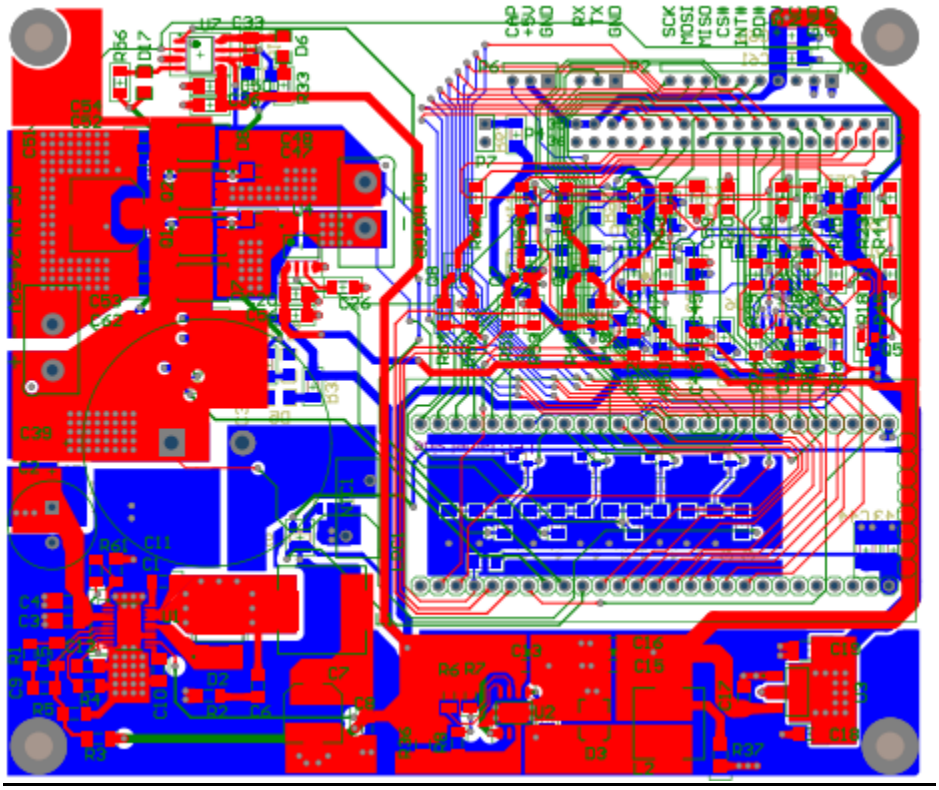

Figura 5-18

A lo largo de todo el diseño (Figura 5-18) se hizo especial énfasis en lograr un diseño robusto frente al ruido, y además se prestó espéciala atención al comportamiento térmico de la placa. En este sentido se buscó distribuir los componentes con gran disipación y se generaron grandes planos a lo largo del PCB para mejorar la disipación.

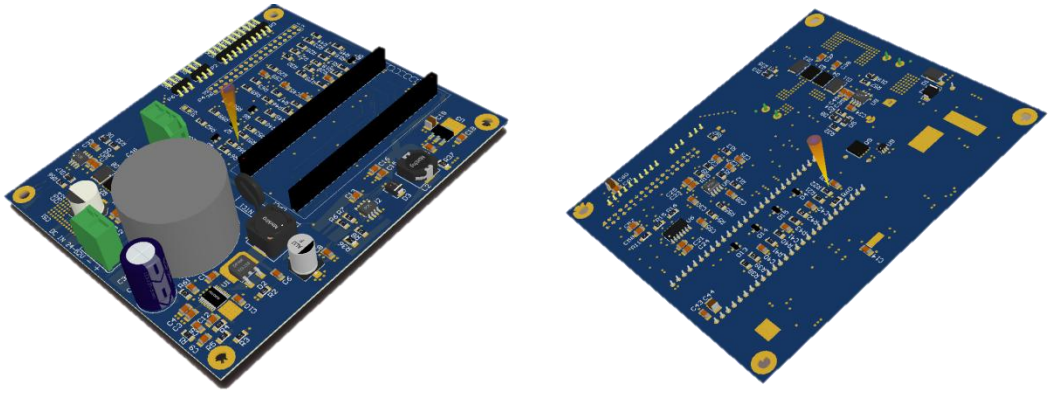

Figura 5-19

En la Figura 5-19 se observan dos imágenes renderizadas 3D del PCB finalizado. A lo largo de todo el desarrollo, en cada componente que se agregó al diseño se realizó un esquemático asociado a un footprint, y un componente en 3d, lo que finalmente permitió hacer un modelo 3d del circuito final, permitiendo esto tener una representación exacta de la ubicación y el espaciado entre los componentes de la placa.

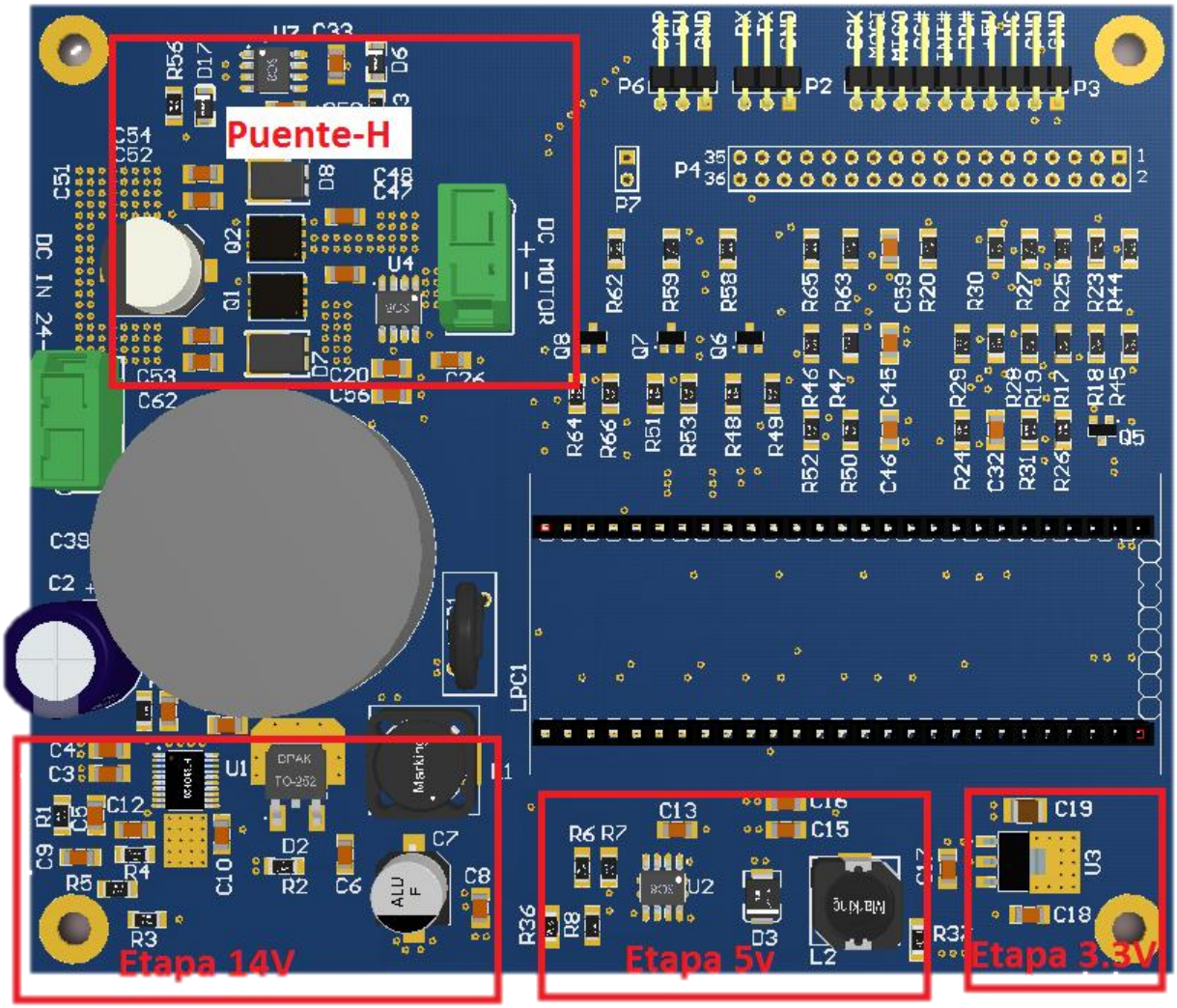

Figura 5-20

La Figura 5-20 muestra la disposición de la capa superior. En la Figura 5-21 se observa a capa inferior.

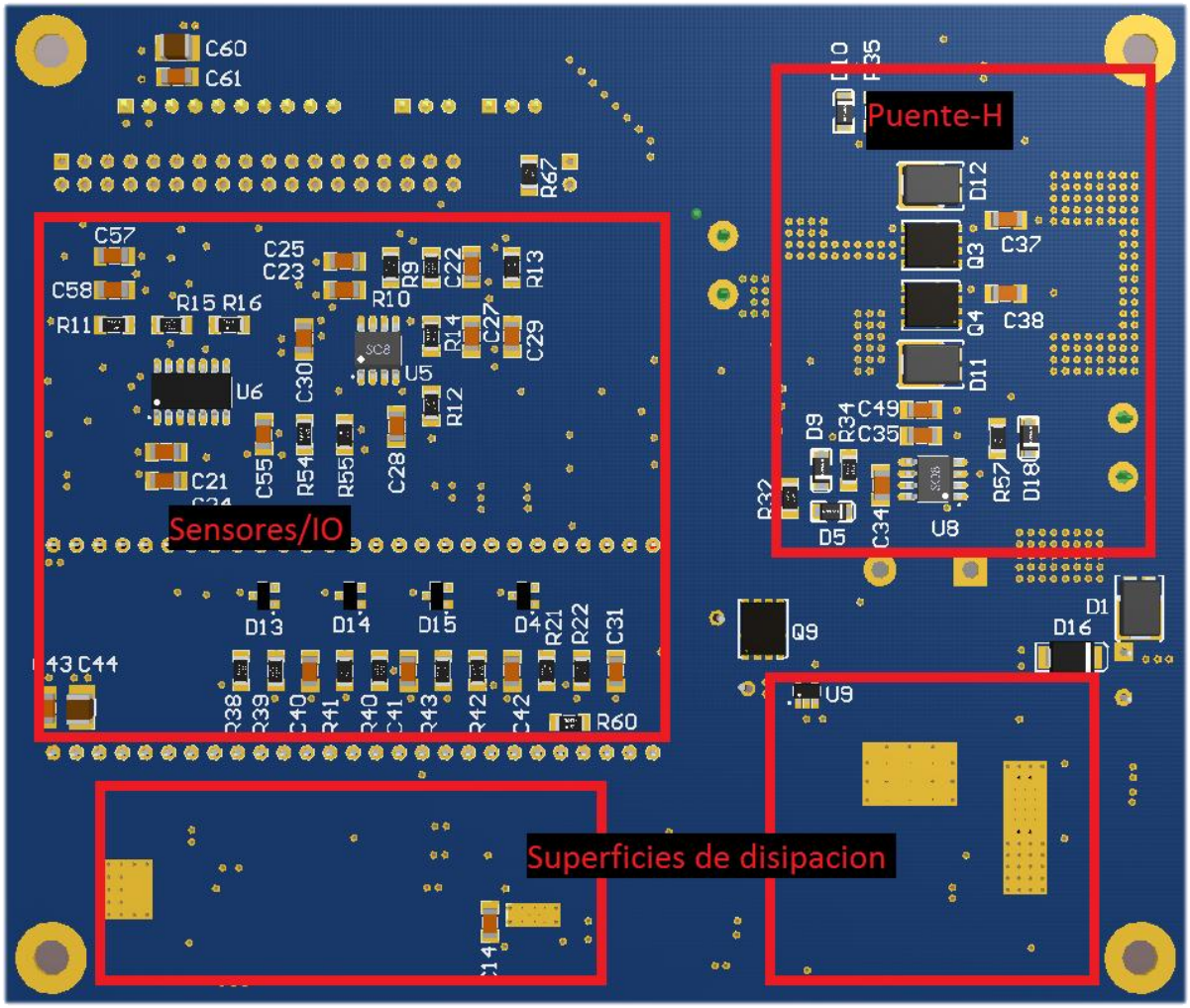
Puente-H
Sensores/IO
Superficies de disipacion
C60
C61
C57
C58
C25
C23
R15 R16
R11
R10
R9
C22
R13
R14
C27
C29
U6
U5
C30
R12
C21
C55
R54
R55
C28
R67
D13
D14
D15
D4
R21
R22
C31
R38
R39
C40
R41
R40
C41
R43
R42
C42
R60
C43 C44
D12
Q3
C37
Q4
C38
D11
D9
R34
C49
C35
R32
D5
C34
U8
R57
D18
Q9
D1
D16
U9

Figura 5-21

6. Desarrollo del Firmware

6.1 Introducción:

Un porcentaje grande del proyecto, corresponde al software desarrollado, dado que por tal medio se resolvieron cuestiones como interfaz HMI, Control de la planta e inicialización de los componentes del circuito.

Para la programación y debugeo del firmware que corre sobe el uC, se utilizó el entorno de desarrollo propio de la marca NXP, LPCXpresso IDE. El mismo es un entorno basado en Eclipse, que integra todas las herramientas de desarrollo que permiten, conjunto con la interfaz JTAG que provee el stick utilizado, programar en C, debugear en tiempo real la aplicación y además integra librerías que facilitan la utilización de los distintos periféricos del uC.

Dado que a lo largo del proyecto el firmware creció más de lo esperado, se comenzó a trabajar en cercanías del límite de los recursos del uC. Por lo tanto fue necesario diseñar una arquitectura de software eficiente, de modo de generar un sistema fluido, capas de administrar los distintos módulos sin generar conflictos ente si.

6.2 Arquitectura de software

El firmware se divide en tres módulos principales (Tabla 6-1):

FT800	Interfaz HMI
Control	Control de motor
Main	Principal/Inicialización

Tabla 6-1

A continuación se desarrollara cada módulo de manera independiente.

6.2.1 FT800

En dicho modulo se encuentra todo el software relacionado con el Display FT800, por lo tanto, aquí es donde se desarrolla todo la interfaz gráfica que se brinda al usuario.

Dentro de FT800 hay tres capas se software, cuya arquitectura se representa en la Tabla 6-1

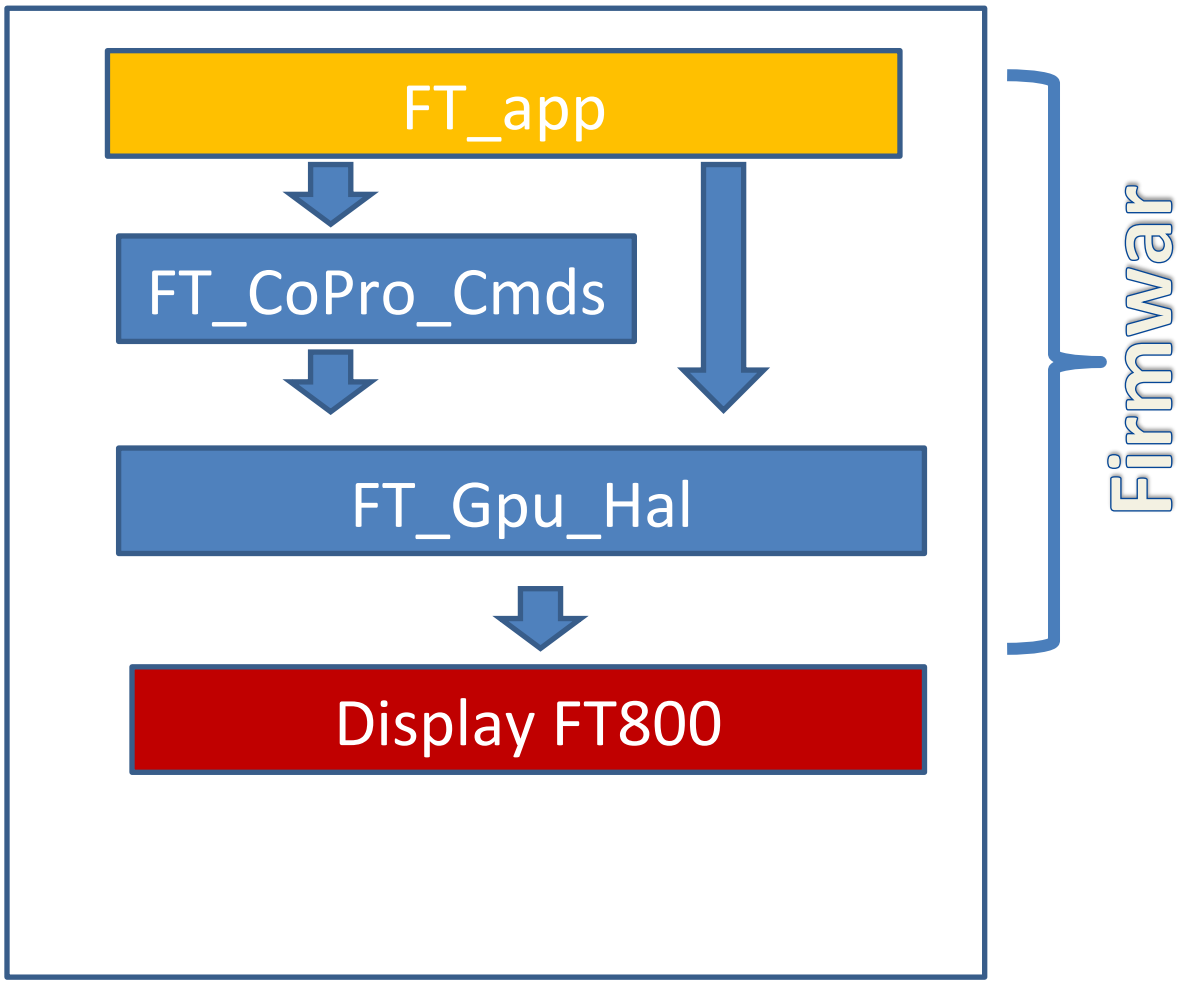

Figura 6-1

Como se observa por encima del controlador de display FT800, se ubica la capa de abstracción de hardware FT_Gpu_Hal. En el diseño del hardware se vio que la conexión entre el CPU y el Display, está dada por una serie de GPIO's (CS, PD) y una conexión serie SPI.

La capa de FT_Gpu_Hal involucra la inicialización de los GPIO's y la inicialización del SPI y provee métodos para abstraer a las capas superiores del uC en el cual está corriendo el firmware. Es decir, para portar el código a otro dispositivo solo es necesario modificar dicha capa.

En este caso, se usaron librerías facilitadas por el fabricante del controlador, pero entre los distintos dispositivos para los cuales estaban realizadas dichas librerías, no se encontraba el LPC1768, ni ningún dispositivo de la firma NXP. Por lo que fue necesario portar el código para el uC LPC1768.

Para hacerlo, como se dijo, fue necesario solo modificar la capa FT_Gpu_Hal. En la Figura 6-2 se da un ejemplo. A lo largo de todo el módulo FT_Gpu_Hal, se encuentran definidas dentro de macros condicionales las sentencias para las distintas plataformas. Lo que se hizo, fue agregar las sentencias equivalentes para el LPC1768 dentro de la macro condicional LPC_PLATAFORM_SPI. Luego se definió dicha macro para que al momento de compilar, dichas sentencias sean agregadas al código final.

```
#if defined(ARDUINO_PLATFORM_SPI) || defined(MSVC_FT800EMU)
                SizeTransfered = 0;
             while (length--) {
                    Ft_Gpu_Hal_Transfer8(host,*buffer);
                buffer++;
                    SizeTransfered ++;
             }
                SizeTransfered = length;
#endif

#ifdef MSVC_PLATFORM_SPI
             {
                 SPI_Write(host-
>hal_handle,buffer,length,&SizeTransfered,SPI_TRANSFER_OPTIONS_SIZE
_IN_BYTES);
                    length = SizeTransfered;
                buffer += SizeTransfered;
             }
#endif

#ifdef LPC_PLATFORM_SPI

             spi_xf.cnt = 0;
             spi_xf.length = length;
             spi_xf.pTxData = buffer;
             spi_xf.pRxData = spi_rx_buf;
             Board_SPI_AssertSSEL();
             Chip_SPI_RWFrames_Blocking(LPC_SPI, &spi_xf);

#endif
```

Figura 6-2(en amarillo se observa el código agregado)

Una vez finalizada la portación del código, fue posible empezar a correr en el uC los ejemplos y librerías provistas por FTDI.

La segunda capa (FT_CoPro_Cmds) también provista por FTDI, contiene métodos para graficar sobre el display determinados Widget y objetos que se encuentran embebidos dentro del controlador FT800.

Llegado este punto se estuvo en condiciones de empezar a programar la interfaz HMI propiamente dicha. Todo el código generado para presentar cada una de las pantallas que se diseñaron está contenido en la capa de aplicación, dentro del módulo FT_app.

Pantallas

Se pensó un esquema de diferentes pantallas por donde el usuario va navegando de manera ascendente, a medida que completa la información necesaria para iniciar un proceso.

Pantalla de inicialización:

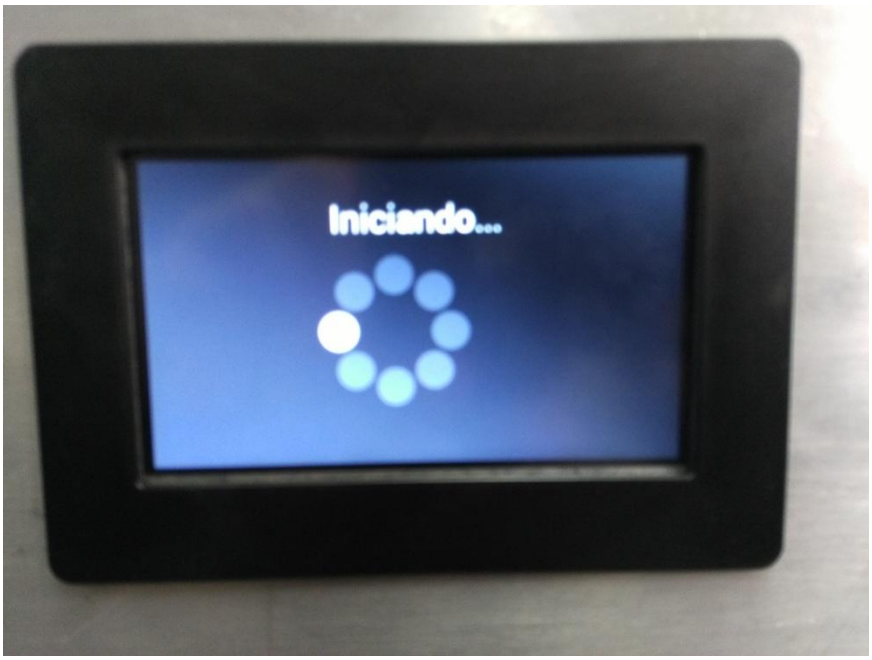

Figura 6-3

En la Figura 6-3 se muestra la pantalla que aparece al momento de encender el equipo. Donde se muestra un widget de “tiempo” durante el transcurso de inicialización del sistema.

Pantalla de inicio

Figura 6-4

La Figura 6-4 se muestra la pantalla en la cual arranca siempre el sistema luego de ser encendió. Cuando el usuario se encuentra en esta pantalla, se le ofrece la posibilidad de iniciar un nuevo proceso, garantizando siempre que todos los datos de un experimento anterior fueron borrados y se arranca un procedimiento completamente nuevo.

Pantalla de ingreso de coordenadas

Figura 6-5

Una vez que el usuario decide comenzar un nuevo proceso, el sistema se posiciona en la pantalla que permite ingresar las coordenadas que conforman el perfil de velocidad que el equipo luego va a reproducir (Figura 6-5).

Mediante el botón agregar se despliega una pantalla que permite el ingreso de las coordenadas una por una.

Como se aprecia en la Figura 6-5, a mediad que el usuario va ingresando coordenadas, las mismas se van listando en un campo que las contiene, y permite desplazarse entre las mismas a través de un scroll-bar. Asociada a cada coordenada, se crean dos botones, "Editar" y "borrar". Los mismos permiten editar la coordenada, o eliminarla respectivamente.

Una vez que el usuario finaliza el ingreso de todos los pares-ordenados, procede a confirmar el perfil de velocidad y aceleración. En este punto se produce una validación de los datos, que evitan que el sistema caiga en cualquier estado indeseado de mal funcionamiento. Si se detectan inconvenientes, un cartel pop-up advierte al usuario y lo devuelve a la misma pantalla permitiéndole modificar las coordenadas que causan el inconveniente. Algunas de las validaciones que se hacen son:

- Debe comenzar en (0,0)
- Debe comenzar con un periodo de aceleración
- El orden debe ser: aceleración, velocidad constante, aceleración, velocidad constante, … ,aceleración.
- La velocidad máxima no debe superar 10000rpm
- La rampa de velocidad máxima no debe superar 4000rpm/s
- Debe finalizar en velocidad cero
- El tiempo debe ser ascendente

Si el perfil de velocidad seteado supera exitosamente el proceso de validación, entonces el sistema pasa a la siguiente pantalla: "Perfil de velocidad a reproducir".

Pantalla de Ingreso de datos

La pantalla que le permite al usuario ingresar datos, consta de un teclado numérico que permite ingresar valores en dos campos: Tiempo y Velocidad. El usuario puede seleccionar mediante el Touchscreen en cual campo desea ingresar datos, mostrándose el campo que está actualmente seleccionado con un borde resaltado, y un cursor titilando donde se imprimirá el próximo carácter. En el caso de la Figura 6-6, se observa seleccionado el campo Tiempo. Cuando el operador se encuentra conforme con los valores ingresados, procede a Aceptar.

Figura 6-6

Pantalla de grafico de perfil a reproducir

En esta pantalla, Figura 6-7, Se grafican las coordenadas ingresadas en el paso previo por el operador. Se realiza un gráfico XY, donde el eje Y corresponde a la velocidad en revoluciones por minutos, y X al tiempo en Segundos.

Al momento de graficar el software hace un auto-ajuste de los imites del gráfico, permitiendo de este modo que sea cual sea el perfil de velocidad seteado, siempre se vea a una escala acorde que permite la fácil visualización del mismo.

El objetivo de esta pantalla es, que el operador pueda confirmar de manera visual, que las coordenadas que ingreso se corresponden con el perfil de velocidad que realmente quiere desarrollar. Dada esta condición, el usuario confirma el perfil, pasando a la última pantalla de la interfaz HMI.

En caso de detectar que no corresponde a lo deseado, se le ofrece la posibilidad de volver a la pantalla anterior para “ajustar” los valores de los pares ordenados.

Figura 6-7

Pantalla Proceso

Figura 6-8

Por último se ubica la pantalla “iniciar Proceso”, (Figura 6-7). La misma consta de:

- Un indicador de RPM símil analógico.
- Una barra de progreso que indica el tiempo transcurrido respecto del tiempo total de duración del proceso.
- Un botón que permite volver a la pantalla anterior
- Un botón que permite iniciar/detener el proceso.

El usuario puede detener el proceso en el momento que lo desee, esto genera una para da inmediata del motor, a la máxima aceleración permitida por el sistema.

6.2.2 Control

Todo el firmware asociado al control del motor se resolvió dentro del módulo “controlador.c”. El código correspondiente a dicho modulo se encuentra en el anexo

Dado que el sensor de velocidad no logro funcionar con precisión en las etapas de aceleración, las rampas de velocidad se lograron con un esquema de lazo abierto, utilizando el modelo del motor. Dicha función se puede ver en la página 46 como Va OpenLoop().

Una vez que el sistema se encuentra en una etapa de velocidad constante, el control conmuta a un esquema de lazo cerrado, donde dicha función se puede ver en la página 46, como Controlador().

ANEXO I – Código Fuente

Controlador.c

```
#include "board.h"
#include "controlador.h"
#include "global.h"
#include "FT_Platform.h"


#define Ts  (0.001)

volatile ref_t refstruct[32];
volatile float feedback_rad = 0;

static float gain(float ref_rad) {
        float Kc, Ki;

        if (ref_rad < 209) {
                Ki = 0.05;
        } else {
                Ki = 0.08;
        }

        Kc = (Ki * Ts / 2);

        return Kc;

}

inline float controlador(float err_old_ini, float u_old_ini, float ref) {

        static float err, err_old, u, u_old, feedback_rad_old = 0;

        feedback_rad = (float) ((24000000.0 / capI) * 6.28);


        err = ref - feedback_rad;

        if (u_old_ini != 0) {

                u_old = u_old_ini;
                err_old = err;

        }

        //integrador
        u = ((gain(ref)) * (err + err_old)) + u_old;

        err_old = err;
        u_old = u;

        if ((feedback_rad > ref * 1.2) || (feedback_rad < ref * 0.8))
                feedback_rad = ref;
        feedback_rad_old = feedback_rad;

        return u;

}
#define Kie  (0.1)
#define Wp   600

volatile int west;
float estimador(float Va, float ref) {
```

```
        static int Ia_old = 0;

        signed int Ia = 1841 - readADC(ADC_VIOUT_CHANNEL);

        west = Va * 222.1
                        + (-(float) Ia * 136.0 - (float) (184.5 * (Ia - Ia_old))) / 131.0;
        Ia_old = Ia;

        if ((west > ref * 1.2) || (west < ref * 0.8))
                return ref;
        return west;

}

//RPM
float VaOpenLoop(float ref) {

        static int Ia_old = 0;
        signed int Ia = 1841 - readADC(ADC_VIOUT_CHANNEL);

        float Va = (ref / 222.22) + ((float) Ia / 1756.75)
                        + ((float) (Ia - Ia_old) / 878.38);

        Ia_old = Ia;
        return Va;

}

typedef struct perfil {
        int RPM;
        int t;
        int tipo;
} perfil_t;

perfil_t perfil[10];

static int time = 0;
static int nroCoo = 1;
volatile int flagEstado = 0;
void cargarRef(void) {

        time = 0;
        nroCoo = 1;

        NVIC_EnableIRQ(RITIMER_IRQn);
}

volatile unsigned int wmeter;
volatile int flagDetener;
void RIT_IRQHandler(void)  //1ms
{
        static float ref = 0;
        static int flagConIni = 0;
        static unsigned int wmetersum = 0, i = 0;
        /* Clearn interrupt */
        Chip_RIT_ClearInt(LPC_RITIMER);

        int tipo = refstruct[nroCoo].tipo;

        if (flagDetener) {
                ref = ref - 2; //-2RPM/ms
                tipo = rampa;

                if (ref < 100) {
                        tensionMOTOR(0);
                        flagDetener = 0;
                        NVIC_DisableIRQ(RITIMER_IRQn);
                        currTime = 0;
                        return;
```

```
                }

        }

        if (((time) < refstruct[nroCoo].timems) && (!flagDetener)) {

                if (refstruct[nroCoo].tipo == constante) {
                        ref = refstruct[nroCoo].val;
                }
                if (refstruct[nroCoo].tipo == rampa) {
                        ref =
                                        (refstruct[nroCoo].val
                                                        * ((float) ((time) - refstruct[nroCoo - 1].timems)
                                                                        / 1000.0)) + refstruct[nroCoo -
1].val;
                }
        } else if (!flagDetener) {
                nroCoo++;
                wmetersum = 0;
        }

        if ((ref <= 950) || (tipo == rampa)) {

                float Va = VaOpenLoop(ref);
                tensionMOTOR(Va);
                flagConIni = 0;
                wmeter = ref; //estimador(Va,ref);

        } else {

                float refrad = (float) ref / 60.0 * 6.28;

                if (flagConIni == 0) {
                        tensionMOTOR(controlador(0, VaOpenLoop(ref), refrad)); //(float)refrad) * 0.043,
                        flagConIni = 1;
                } else {
                        tensionMOTOR(controlador(0, 0, refrad));

                }

                wmeter = (int) feedback_rad / 6.28 * 60.0;

        }

        currTime = time;
        time++;
        if (((time) > refstruct[nroCoo].timems)
                        & (refstruct[nroCoo + 1].val == 0)) {
                flagEstado = 0;
                currTime = 0;
                tensionMOTOR(0);
                NVIC_DisableIRQ(RITIMER_IRQn);

        }

}
```

FT_app.c

```
//*************************************************************************************************************
*****************************************************************************
void borrarCoordeanda(contextoPantalla *ptrContexto) {

        int i = (ptrContexto->nroCoordenadas - ptrContexto->currIdx);
        int j = ptrContexto->currIdx;
        while (j < ptrContexto->nroCoordenadas) {
```

```
                ptrContexto->vec_Coordenadas[j].time_int =
                                ptrContexto->vec_Coordenadas[j + 1].time_int;
                ptrContexto->vec_Coordenadas[j].RPM_int = ptrContexto->vec_Coordenadas[j
                                + 1].RPM_int;
                strcpy(ptrContexto->vec_Coordenadas[j].RPM_Array,
                                ptrContexto->vec_Coordenadas[j + 1].RPM_Array);
                strcpy(ptrContexto->vec_Coordenadas[j].time_Array,
                                ptrContexto->vec_Coordenadas[j + 1].time_Array);

                j++;
        }
        (ptrContexto->nroCoordenadas)--;
        ptrContexto->vec_Coordenadas[j].RPM_Array[0] = '\0';
        ptrContexto->vec_Coordenadas[j].time_Array[0] = '\0';
        ptrContexto->vec_Coordenadas[j].time_int = 0;
        ptrContexto->vec_Coordenadas[j].RPM_int = 0;
        ptrContexto->vec_Coordenadas[j + 1].time_int = 0;
        ptrContexto->vec_Coordenadas[j + 1].RPM_int = 0;

}
void numsound(int tag) {

        int value = ((tag - 48) & (0x000F)) | (0x0030);

        Ft_Gpu_Hal_Wr8(phost, REG_VOL_SOUND, 100);
        Ft_Gpu_Hal_Wr16(phost, REG_SOUND, value);
        Ft_Gpu_Hal_Wr8(phost, REG_PLAY, 1);

}
void buttonsound(void) {
        Ft_Gpu_Hal_Wr8(phost, REG_VOL_SOUND, 80);
        Ft_Gpu_Hal_Wr16(phost, REG_SOUND, 0x0051);
        Ft_Gpu_Hal_Wr8(phost, REG_PLAY, 1);
}
void mute(void) {

        Ft_Gpu_Hal_Wr8(phost, REG_VOL_SOUND, 0);
        Ft_Gpu_Hal_Wr16(phost, REG_SOUND, 0);
        Ft_Gpu_Hal_Wr8(phost, REG_PLAY, 1);
}

void errorsound(void) {

        Ft_Gpu_Hal_Wr8(phost, REG_VOL_SOUND, 250);
        Ft_Gpu_Hal_Wr16(phost, REG_SOUND, (0x1E));
        Ft_Gpu_Hal_Wr8(phost, REG_PLAY, 1);

}

void pantallaIngresarDatos(contextoPantalla *ptrContexto) {
#define boton10  10
#define boton11  11
#define boton12  12
#define Disp_t   13
#define Disp_RPM 14

        static ft_int16_t TextFont = 29, ButtonW = 30, ButtonH = 53, yBtnDst = 3,
                        yOffset, xOffset;
        static ft_char8_t DispText_t[10] = { '\0' }, DispText_RPM[10] = { '\0' },
                        CurrChar = '|';
        static ft_uint8_t CurrTag = 0, Pendown = 1, cntCursor = 0;
        static ft_int32_t CurrTextIdx, tagoptionBtn10, tagoptionBtn11,
                        tagoptionBtn12 = 0;
        static int currDispText = 0;

        ft_char8_t StringArray[10];

        if (ptrContexto->flagborrarDisp == 1) {
                DispText_t[0] = '\0';
```

```
        DispText_RPM[0] = '\0';
        CurrTextIdx = 0;
        ptrContexto->flagborrarDisp = false;

        if (ptrContexto->modo == editar) {
                int j;
                j = (ptrContexto->currIdx);
                strcat(DispText_RPM, ptrContexto->vec_Coordenadas[j].RPM_Array);
                strcat(DispText_t, ptrContexto->vec_Coordenadas[j].time_Array);
                currDispText = 1;
                CurrTextIdx = strlen(DispText_RPM);
        }
}

/* Check the user input and then add the characters into array */
CurrTag = Ft_Gpu_Hal_Rd8(phost, REG_TOUCH_TAG);

Pendown = ((Ft_Gpu_Hal_Rd32(phost, REG_TOUCH_DIRECT_XY) >> 31) & 0x01);

CurrChar = CurrTag;
if ((CurrTag <= 45) || (CurrTag >= 60)) {
        cntCursor++;
        if (cntCursor <= 25)
                CurrChar = '|';
        if (cntCursor > 25)
                CurrChar = ' ';
        if (cntCursor == 50)
                cntCursor = 0;
}

if (1 == Pendown) {
        tagoptionBtn10 = 0;
        tagoptionBtn11 = 0;
        tagoptionBtn12 = 0;
        // mute();
}
if (Pendown == 0) {
        switch (CurrTag) {
        case boton10:
                tagoptionBtn10 = OPT_FLAT;
                break;
        case boton11:
                tagoptionBtn11 = OPT_FLAT;
                break;
        case boton12:
                tagoptionBtn12 = OPT_FLAT;
                break;

        }

}
/* check whether pwndown has happened */
if ((CurrTag != ptrContexto->PrevTag)) {

        /* check whether pwndown has happened */
        if ((1 == Pendown) && (ptrContexto->PrevTag > 45)
                        && (ptrContexto->PrevTag < 60)) {
                CurrTextIdx++;
                if (CurrTextIdx == 7) {
                        CurrTextIdx = 6;
                }

        }
        if ((Pendown == 0) && (CurrTag > 45) && (CurrTag < 60))
                buttonsound();  //numsound(CurrTag);

        switch (ptrContexto->PrevTag) {
        case boton10:
                //   tagoptionBtn10 = OPT_FLAT;
```

```
        if ((CurrTextIdx == 0) && (currDispText == 1)) {
                CurrTextIdx = strlen(DispText_t) + 1;
                currDispText = 0;
                DispText_RPM[0] = '\0';
        }

        if (CurrTextIdx > 0)
                CurrTextIdx--;

        buttonsound();

        break;
case boton11:
        //         tagoptionBtn11 = OPT_FLAT;
        buttonsound();

        if (currDispText == 0) {
                if (DispText_t[1] == '\0')
                        goto retorno;

                currDispText = 1;
                DispText_t[CurrTextIdx] = '\0';
                CurrTextIdx = strlen(DispText_RPM);

        } else if (currDispText == 1) {
                if (DispText_RPM[1] == '\0')
                        goto retorno;
                if (DispText_t[0] == '\0')
                        goto retorno;

                int i;

                if (ptrContexto->modo == agregar)
                        i = (ptrContexto->nroCoordenadas)++;
                if (ptrContexto->modo == editar)
                        i = (ptrContexto->currIdx);

                ptrContexto->vec_Coordenadas[i].RPM_Array[0] = '\0';
                ptrContexto->vec_Coordenadas[i].time_Array[0] = '\0';

                DispText_RPM[CurrTextIdx] = '\0';
                strcat(ptrContexto->vec_Coordenadas[i].RPM_Array, DispText_RPM);
                strcat(ptrContexto->vec_Coordenadas[i].time_Array, DispText_t);

                ptrContexto->vec_Coordenadas[i].RPM_int = atoi(DispText_RPM);
                ptrContexto->vec_Coordenadas[i].time_int = (atoi(DispText_t)
                                * 1000);

                currDispText = 0;

                selectPantalla = pantallaMenuPrincipalActiva;
                return;

        }
        break;

case boton12:
        currDispText = 0;
        buttonsound();
        selectPantalla = pantallaMenuPrincipalActiva;
        return;
        break;

case Disp_t:
        if (currDispText) {
                currDispText = 0;
                DispText_RPM[CurrTextIdx] = '\0';
                CurrTextIdx = strlen(DispText_t);
```

```
                }

                break;
        case Disp_RPM:
                if (!currDispText) {
                        currDispText = 1;
                        DispText_t[CurrTextIdx] = '\0';
                        CurrTextIdx = strlen(DispText_RPM);

                }
                break;

        }

}

/* Draw text entered by user */
/* make sure the array is a string */

retorno: if (CurrTextIdx < 6) {
        if (currDispText == 0) {
                DispText_t[CurrTextIdx] = CurrChar;
                DispText_t[CurrTextIdx + 1] = '\0';
        } else if (currDispText == 1) {
                DispText_RPM[CurrTextIdx] = CurrChar;
                DispText_RPM[CurrTextIdx + 1] = '\0';
        }
}

Ft_Gpu_CoCmd_Dlstart(phost);
Ft_App_WrCoCmd_Buffer(phost, CLEAR_COLOR_RGB(32, 32, 32));
Ft_App_WrCoCmd_Buffer(phost, CLEAR(1, 1, 1));

Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(64, 64, 64));
Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(10 * 16)); //LINE_WIDTH is used for corner curvature
Ft_App_WrCoCmd_Buffer(phost, BEGIN(RECTS));
Ft_App_WrCoCmd_Buffer(phost, VERTEX2F((20) * 16, (38) * 16));
Ft_App_WrCoCmd_Buffer(phost, VERTEX2F((300) * 16, (245) * 16));

//cuadro de texto
#define cuadrotxt_x 165
#define cuadrotxt_y 55
#define cuadrotxt_w 120
#define cuadrotxt_h 20

Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(32, 32, 32));
Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(10 * 16)); //LINE_WIDTH is used for corner curvature

Ft_App_WrCoCmd_Buffer(phost, BEGIN(RECTS));
Ft_App_WrCoCmd_Buffer(phost,
                VERTEX2F((cuadrotxt_x-4)*16,
                                (cuadrotxt_y - 4 + currDispText * 70)*16));
Ft_App_WrCoCmd_Buffer(phost,
                VERTEX2F((cuadrotxt_x+cuadrotxt_w + 4)*16,
                                (cuadrotxt_y + cuadrotxt_h + 4 + currDispText * 70)*16));

Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(1));

Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(220, 220, 220));

Ft_App_WrCoCmd_Buffer(phost, TAG(Disp_t));
Ft_App_WrCoCmd_Buffer(phost, BEGIN(RECTS));
Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(cuadrotxt_x*16, cuadrotxt_y*16));
Ft_App_WrCoCmd_Buffer(phost,
                VERTEX2F((cuadrotxt_x+cuadrotxt_w)*16,
                                (cuadrotxt_y+cuadrotxt_h)*16));

Ft_App_WrCoCmd_Buffer(phost, TAG(Disp_RPM));
Ft_App_WrCoCmd_Buffer(phost, BEGIN(RECTS));
```

```
Ft_App_WrCoCmd_Buffer(phost,
                VERTEX2F(cuadrotxt_x*16, (cuadrotxt_y + 70)*16));
Ft_App_WrCoCmd_Buffer(phost,
                VERTEX2F((cuadrotxt_x+cuadrotxt_w)*16,
                                (cuadrotxt_y+cuadrotxt_h+70)*16));

Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(0));

//Barra sup
Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(64, 64, 64));
Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(1 * 16)); //LINE_WIDTH is used for corner curvature
Ft_App_WrCoCmd_Buffer(phost, BEGIN(RECTS));
Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(0 * 16, 0 * 16));
Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(480 * 16, 20 * 16));

Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0, 0, 0));
Ft_App_WrCoCmd_Buffer(phost, BEGIN(LINES));
Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(1 * 16)); //LINE_WIDTH is used for corner curvature
Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(0 * 16, 21 * 16));
Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(480 * 16, 21 * 16));
Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 255));
Ft_Gpu_CoCmd_Text(phost, FT_DispWidth / 2, 10, 27, OPT_CENTER,
                "Ingrese un par tiempo-velocidad");

Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0xff, 0xff, 0xff));
Ft_Gpu_CoCmd_Text(phost, cuadrotxt_x - 150, cuadrotxt_y, 27, 0,
                "Tiempo(seg)");
Ft_Gpu_CoCmd_Text(phost, cuadrotxt_x - 150, cuadrotxt_y + 70, 27, 0,
                "Velocidad(RPM)");

Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0x00, 0x00, 0x00));
Ft_Gpu_CoCmd_Text(phost, cuadrotxt_x + 5, cuadrotxt_y, TextFont, 0,
                DispText_t);  //text
Ft_Gpu_CoCmd_Text(phost, cuadrotxt_x + 5, cuadrotxt_y + 70, TextFont, 0,
                DispText_RPM);  //text

Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(1));

Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0xff, 0xff, 0xff));
Ft_Gpu_CoCmd_FgColor(phost, 0x404080);
Ft_Gpu_CoCmd_GradColor(phost, 0x00ff00);
Ft_Gpu_CoCmd_BgColor(phost, 0x0000ff);

yOffset = 30;
Ft_Gpu_CoCmd_Keys(phost, 320, yOffset, 150, ButtonH, TextFont,
                (0 | CurrTag), "789");
yOffset += ButtonH + yBtnDst;
Ft_Gpu_CoCmd_Keys(phost, 320, yOffset, 150, ButtonH, TextFont,
                (0 | CurrTag), "456");
yOffset += ButtonH + yBtnDst;
Ft_Gpu_CoCmd_Keys(phost, 320, yOffset, 150, ButtonH, TextFont,
                (0 | CurrTag), "123");
yOffset += ButtonH + yBtnDst;
// Ft_App_WrCoCmd_Buffer(phost,COLOR_A(150));

Ft_Gpu_CoCmd_Keys(phost, 397, yOffset, 73, ButtonH, TextFont, (0 | CurrTag),
                "0");

Ft_App_WrCoCmd_Buffer(phost, TAG(boton10));
Ft_Gpu_CoCmd_Button(phost, 320, yOffset, 73, ButtonH, TextFont,
                tagoptionBtn10, "<-");

Ft_Gpu_CoCmd_FgColor(phost, 0x009900);
Ft_App_WrCoCmd_Buffer(phost, TAG(boton11));
Ft_Gpu_CoCmd_Button(phost, 170, 180, 100, 50, 29, tagoptionBtn11,
                "Aceptar");

Ft_Gpu_CoCmd_FgColor(phost, 0xBA0000);
Ft_App_WrCoCmd_Buffer(phost, TAG(boton12));
```

```
        Ft_Gpu_CoCmd_Button(phost, 50, 180, 100, 50, 29, tagoptionBtn12,
                        "Cancelar");

        Ft_App_WrCoCmd_Buffer(phost, DISPLAY());
        Ft_Gpu_CoCmd_Swap(phost);

        /* Download the commands into fifo */
        Ft_App_Flush_Co_Buffer(phost);

        /* Wait till coprocessor completes the operation */
        Ft_Gpu_Hal_WaitCmdfifo_empty(phost);

        ptrContexto->PrevTag = CurrTag;

}

void pantallaGrafico(contextoPantalla *ptrContexto) {
#define W_cuadro 330
#define h_cuadro 200
#define x_cuadro 50
#define y_cuadro 40

#define boton1 1
#define boton2 2
#define boton3 3

        static ft_int16_t TextFont = 29, ButtonW = 30, ButtonH = 53, yBtnDst = 3,
                        yOffset, xOffset;
        static ft_char8_t DispText[10], CurrChar = '|';
        static ft_uint8_t CurrTag = 0, Pendown = 1, cntCursor = 0;
        static ft_int32_t CurrTextIdx, tagoptionBtn1, tagoptionBtn2, tagoptionBtn3 =
                        0, tagoptionvec[40];

        ft_char8_t StringArray[20];

        /* Check the user input and then add the characters into array */
        CurrTag = Ft_Gpu_Hal_Rd8(phost, REG_TOUCH_TAG);
        Pendown = ((Ft_Gpu_Hal_Rd32(phost, REG_TOUCH_DIRECT_XY) >> 31) & 0x01);

        /* check whether pwndown has happened */
        if ((CurrTag != ptrContexto->PrevTag)) {

                if ((Pendown == 0)) {

                        switch (CurrTag) {

                        case boton1:
                                tagoptionBtn1 = OPT_FLAT;
                                break;
                        case boton2:
                                tagoptionBtn2 = OPT_FLAT;
                                break;
                        case boton3:
                                tagoptionBtn3 = OPT_FLAT;
                                break;

                        }
                }
                if ((Pendown == 1)) {

                        switch (ptrContexto->PrevTag) {

                        case boton1:
                                buttonsound();

                                tagoptionBtn1 = 0;
                                selectPantalla = pantallaProcesoActiva;
                                return;
```

```
                        break;
                case boton2:
                        tagoptionBtn2 = 0;

                        break;
                case boton3:
                        buttonsound();

                        tagoptionBtn3 = 0;

                        selectPantalla = pantallaMenuPrincipalActiva;
                        return;
                        break;

                }
        }

}

/* Draw text entered by user */
/* make sure the array is a string */

Ft_Gpu_CoCmd_Dlstart(phost);
Ft_App_WrCoCmd_Buffer(phost, CLEAR_COLOR_RGB(32, 32, 32));
Ft_App_WrCoCmd_Buffer(phost, CLEAR(1, 1, 1));

Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(0));

//Dibujar Barra superiot*****************************************************************

Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(64, 64, 64));
Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(1 * 16)); //LINE_WIDTH is used for corner curvature
Ft_App_WrCoCmd_Buffer(phost, BEGIN(RECTS));
Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(0 * 16, 0 * 16));
Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(480 * 16, 20 * 16));

Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0, 0, 0));
Ft_App_WrCoCmd_Buffer(phost, BEGIN(LINES));
Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(1 * 16)); //LINE_WIDTH is used for corner curvature
Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(0 * 16, 21 * 16));
Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(480 * 16, 21 * 16));
Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 255));
Ft_Gpu_CoCmd_Text(phost, FT_DispWidth / 2, 10, 27, OPT_CENTER,
                "Perfil de velocidad a reproducir");

//cuadro ********************************************************************************

Ft_App_WrCoCmd_Buffer(phost, SCISSOR_XY(x_cuadro, y_cuadro));
Ft_App_WrCoCmd_Buffer(phost, SCISSOR_SIZE(W_cuadro, h_cuadro));

Ft_App_WrCoCmd_Buffer(phost, CLEAR_COLOR_RGB(255, 255, 255));

Ft_App_WrCoCmd_Buffer(phost, CLEAR(1, 1, 1));

//grafico*************************************************************************************

Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0, 0, 0));
Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(1 * 16)); //LINE_WIDTH is used for corner curvature
Ft_App_WrCoCmd_Buffer(phost, BEGIN(LINES));

int i = 0;
for (i = 0; i < 5; i++) {

        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F((i*W_cuadro/5 + x_cuadro)*16, 0));
        Ft_App_WrCoCmd_Buffer(phost,
                        VERTEX2F((i*W_cuadro/5 + x_cuadro)*16, 270 * 16));

}
```

```
i = 0;
for (i = 0; i < 5; i++) {

        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(0, (i*h_cuadro/5 + y_cuadro)*16));
        Ft_App_WrCoCmd_Buffer(phost,
                        VERTEX2F(480 * 16, (i*h_cuadro/5 + y_cuadro)*16));

}

Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 0, 0));
Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(2 * 16)); //LINE_WIDTH is used for corner curvature
Ft_App_WrCoCmd_Buffer(phost, BEGIN(LINE_STRIP));

int timebase = ptrContexto->t_max;
int cnt = 0;
while (timebase >= 10) {
        timebase = timebase / 10;
        cnt++;
}

if (timebase < 2) {
        timebase = 2;
} else if (timebase < 5) {
        timebase = 5;
} else if (timebase < 10) {
        timebase = 10;
}
while (cnt > 0) {
        timebase = timebase * 10;
        cnt--;
}

int RPMbase = (ptrContexto->RPM_max);
cnt = 0;
while (RPMbase >= 10) {
        RPMbase = RPMbase / 10;
        cnt++;
}

if (RPMbase < 2) {
        RPMbase = 2;
} else if (RPMbase < 5) {
        RPMbase = 5;
} else if (RPMbase < 10) {
        RPMbase = 10;
}

while (cnt > 0) {
        RPMbase = RPMbase * 10;
        cnt--;
}

for (i = 0; (i < (ptrContexto->nroCoordenadas)); i++) {
        int x, y;
        x = ptrContexto->vec_Coordenadas[i].time_int;
        y = ptrContexto->vec_Coordenadas[i].RPM_int;
        //escala

        x = x * W_cuadro / timebase;
        y = y * h_cuadro / RPMbase;

        //referncia al (0,0)
        x = x + x_cuadro;
        y = (h_cuadro + y_cuadro) - y;

        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(x * 16, y * 16));

}
```

```
        Ft_App_WrCoCmd_Buffer(phost, SCISSOR_XY(0, 0));
        Ft_App_WrCoCmd_Buffer(phost, SCISSOR_SIZE(480, 270));

        //*****************************************************************************************
***

        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0xff, 0xff, 0xff));

        float timediv = (((float) timebase / 1000.0) / 5.0);
        for (i = 0; i < 5; i++) {
                sprintf(StringArray, "%.1f", timediv * (float) i);
                Ft_Gpu_CoCmd_Text(phost, x_cuadro + (i * (W_cuadro / 5)),
                                y_cuadro + h_cuadro + 10, 26, OPT_CENTER, StringArray); //text info
        }
        strcpy(StringArray, "[Seg]");
        Ft_Gpu_CoCmd_Text(phost, x_cuadro + W_cuadro, y_cuadro + h_cuadro + 10, 26,
                        OPT_CENTER, StringArray);  //text info

        int RPMdiv = (RPMbase / 5);
        for (i = 0; i < 5; i++) {
                sprintf(StringArray, "%00000d", RPMdiv * i);
                Ft_Gpu_CoCmd_Text(phost, 2, y_cuadro + h_cuadro - (h_cuadro / 5) * i,
                                26, OPT_CENTERY, StringArray);  //text info
        }
        strcpy(StringArray, "[RPM]");
        Ft_Gpu_CoCmd_Text(phost, 2, y_cuadro, 26, OPT_CENTERY, StringArray); //text info

        //*****************************************************************************************
****

        Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(1));

        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0xff, 0xff, 0xff));
        Ft_Gpu_CoCmd_FgColor(phost, 0x404080);
        Ft_Gpu_CoCmd_GradColor(phost, 0x00ff00);

        Ft_Gpu_CoCmd_FgColor(phost, 0x009900);
        Ft_App_WrCoCmd_Buffer(phost, TAG(boton1));
        Ft_Gpu_CoCmd_Button(phost, 390, 50, 80, 70, 27, tagoptionBtn1, "Confirmar");

        Ft_Gpu_CoCmd_FgColor(phost, 0x404080);
        Ft_App_WrCoCmd_Buffer(phost, TAG(boton3));
        Ft_Gpu_CoCmd_Button(phost, 390, 140, 80, 70, 27, tagoptionBtn3, "Volver");

        Ft_App_WrCoCmd_Buffer(phost, DISPLAY());
        Ft_Gpu_CoCmd_Swap(phost);

        /* Download the commands into fifo */
        Ft_App_Flush_Co_Buffer(phost);

        /* Wait till coprocessor completes the operation */
        Ft_Gpu_Hal_WaitCmdfifo_empty(phost);

        ptrContexto->PrevTag = CurrTag;

}

int flagMsg = 0;
char stringMsg[100];
Status chekCoordenadas(contextoPantalla *ptrContexto) {
        int i = 0;
        for (i = 0; i < (ptrContexto->nroCoordenadas - 1); i++) {

                if (ptrContexto->vec_Coordenadas[i].RPM_int
                                != ptrContexto->vec_Coordenadas[i + 1].RPM_int) { // entonces ref
variable, val es la pendiente en RPM/seg

                        refstruct[i + 1].val =
                                        ((float) (ptrContexto->vec_Coordenadas[i + 1].RPM_int
```

```
                                                        - ptrContexto->vec_Coordenadas[i].RPM_int)
                                                        / ((float) (ptrContexto->vec_Coordenadas[i +
1].time_int
                                                                        - ptrContexto-
>vec_Coordenadas[i].time_int)
                                                                        / 1000));
                        refstruct[i + 1].timems =
                                        ptrContexto->vec_Coordenadas[i + 1].time_int;
                        refstruct[i + 1].tipo = rampa;

                } else { //velocidad fija, val es la velocidad en RPM
                        refstruct[i + 1].val = ptrContexto->vec_Coordenadas[i].RPM_int;
                        refstruct[i + 1].timems =
                                        ptrContexto->vec_Coordenadas[i + 1].time_int;
                        refstruct[i + 1].tipo = constante;

                }

        }

        refstruct[i + 1].val = 0;
        refstruct[i + 1].timems = 0;
        refstruct[i + 2].val = 0;
        refstruct[i + 2].timems = 0;
        //chekeo

        //tiempo ascendente
        for (i = 0; i < (ptrContexto->nroCoordenadas - 1); i++) {

                if (ptrContexto->vec_Coordenadas[i + 1].time_int
                                <= ptrContexto->vec_Coordenadas[i].time_int) {
                        strcpy(stringMsg, "El vector de tiempo debe ser ascendente");
                        flagMsg = 0;
                        return ERROR;

                }

        }
        //mas de tres coordenads
        if (ptrContexto->nroCoordenadas < 3) {
                strcpy(stringMsg, "Debe ingresar un minimo de 4 coordenadas");
                flagMsg = 1;
                return ERROR;
        }
        //empieza en cero
        if (ptrContexto->vec_Coordenadas[0].RPM_int != 0) {
                strcpy(stringMsg, "La velocidad inicial debe ser 0 RPM");
                flagMsg = 1;
                return ERROR;
        }
        if (ptrContexto->vec_Coordenadas[0].time_int != 0) {
                strcpy(stringMsg, "El tiempo inicial debe ser 0 seg");
                flagMsg = 1;
                return ERROR;
        }
        //termina en cero
        if (ptrContexto->vec_Coordenadas[ptrContexto->nroCoordenadas - 1].RPM_int
                        != 0) {
                strcpy(stringMsg, "La velocidad final debe ser 0 RPM");
                flagMsg = 1;
                return ERROR;
        }

        //los pares tiene q ser aceleracion, impares constante
        for (i = 0; i < (ptrContexto->nroCoordenadas - 1); i += 2) {

                if (refstruct[i + 1].tipo != rampa) {
                        strcpy(stringMsg, "Perfil de velocidad incompatible");
                        flagMsg = 1;
```

```
                        return ERROR;
                }
                //si es rampa, verifico limite de aceleracion
                if ((refstruct[i + 1].val > aceleracionMAX)
                                || (refstruct[i + 1].val < -aceleracionMAX)) {
                        strcpy(stringMsg, "Excedio la aceleracion maxima posible");
                        flagMsg = 1;
                        return ERROR;
                }

        }
        for (i = 1; i < (ptrContexto->nroCoordenadas - 1); i += 2) {
                if (refstruct[i + 1].tipo != constante) {
                        strcpy(stringMsg, "Perfil de velocidad incompatible");
                        flagMsg = 1;
                        return ERROR;
                }
                //si es cte, verifico vel MAX
                if ((refstruct[i + 1].val) > velocidadMAX) {
                        strcpy(stringMsg, "Excedio la velocidad maxima posible");
                        flagMsg = 1;
                        return ERROR;
                }
                //verifico velocidadminima
                if ((refstruct[i + 1].val) < velocidadMIN) {
                        strcpy(stringMsg, "Debe superar la velocidad minima permitida");
                        flagMsg = 1;
                        return ERROR;
                }

        }

        return SUCCESS;
}

void pantallaMenuPrincipal(contextoPantalla *ptrContexto) {
        static ft_int16_t TextFont = 29, ButtonW = 30, ButtonH = 53, yBtnDst = 3,
                        yOffset, xOffset;
        static ft_char8_t DispText[10], CurrChar = '|';
        static ft_uint8_t CurrTag = 0, Pendown = 1, cntCursor = 0;
        static ft_int32_t CurrTextIdx, tagoptionBtn100, tagoptionBtn101,
                        tagoptionBtn102 = 0, tagoptionBtn104 = 0, tagoptionvec[40];
#define boton100 100
#define boton101 101
#define boton102 102
#define boton104 104
#define scrollTAG 103
        ft_char8_t StringArray[20];

//startHMI = 0;

        /* Check the user input and then add the characters into array */
        CurrTag = Ft_Gpu_Hal_Rd8(phost, REG_TOUCH_TAG);

        Pendown = ((Ft_Gpu_Hal_Rd32(phost, REG_TOUCH_DIRECT_XY) >> 31) & 0x01);

        /* check whether pwndown has happened */

        if (1 == Pendown) {
                tagoptionBtn100 = 0;
                tagoptionBtn101 = 0;
                tagoptionBtn102 = 0;
                tagoptionBtn104 = 0;

                int i = 0;
                while (i < 40) {
                        tagoptionvec[i] = 0;
                        i++;
                }
```

```
}
if (Pendown == 0) {
        switch (CurrTag) {
        case boton100:
                tagoptionBtn100 = OPT_FLAT;
                break;
        case boton101:
                tagoptionBtn101 = OPT_FLAT;
                break;
        case boton102:
                tagoptionBtn102 = OPT_FLAT;
                break;
        case boton104:
                tagoptionBtn104 = OPT_FLAT;
                break;

        }
        if ((CurrTag >= 20) && (CurrTag <= 40)) {
                tagoptionvec[CurrTag - 20] = OPT_FLAT;
                //ptrContexto->currIdx = CurrTag - 20;
        }
        if ((CurrTag >= 60) && (CurrTag <= 80)) {
                tagoptionvec[CurrTag - 40] = OPT_FLAT;
                //ptrContexto->currIdx = CurrTag - 60;
        }
}

if ((CurrTag != ptrContexto->PrevTag)) {
        /* check whether pwndown has happened */
        if ((1 == Pendown)) {
                switch (ptrContexto->PrevTag) {

                case boton100:
                        buttonsound();

                        selectPantalla = pantallaInicioActiva;

                        return;

                        break;
                case boton101:
                        ;

                        buttonsound();

                        int i = ptrContexto->nroCoordenadas;
                        int MAX = ptrContexto->vec_Coordenadas[i - 1].time_int;
                        if (i == 0)
                                MAX = 0;
                        if (i == 1)
                                MAX = ptrContexto->vec_Coordenadas[0].time_int;
                        while (i > 1) {
                                if (ptrContexto->vec_Coordenadas[i - 2].time_int > MAX) {
                                        MAX = ptrContexto->vec_Coordenadas[i - 2].time_int;
                                }
                                i--;
                        }
                        ptrContexto->t_max = MAX;
                        i = ptrContexto->nroCoordenadas;
                        MAX = ptrContexto->vec_Coordenadas[i - 1].RPM_int;
                        if (i == 0)
                                MAX = 0;
                        if (i == 1)
                                MAX = ptrContexto->vec_Coordenadas[0].RPM_int;
                        while (i > 1) {
                                if (ptrContexto->vec_Coordenadas[i - 2].RPM_int > MAX) {
                                        MAX = ptrContexto->vec_Coordenadas[i - 2].RPM_int;
                                }
```

```
                        i--;
                }
                ptrContexto->RPM_max = MAX;

                if (!chekCoordenadas(ptrContexto)) {
                        flagMsg = 1;
                        errorsound();
                        goto retorno;

                }

                selectPantalla = pantallaGraficoActiva;

                return;
                break;
        case boton102:
                buttonsound();

                if (ptrContexto->nroCoordenadas >= 20) {
                        strcpy(stringMsg, "No es posible ingresar mas coordenadas");
                        flagMsg = 1;
                        errorsound();

                        goto retorno;
                }
                ptrContexto->modo = agregar;
                ptrContexto->flagborrarDisp = true;
                selectPantalla = pantallaIngresarDatosActiva;
                return;
                break;
        case boton104:
                buttonsound();

                flagMsg = 0;
                break;
        }

        retorno: if ((ptrContexto->PrevTag >= 20)
                        && (ptrContexto->PrevTag <= 40)) {
                ptrContexto->currIdx = ptrContexto->PrevTag - 20;
                ptrContexto->modo = editar;
                ptrContexto->flagborrarDisp = true;
                selectPantalla = pantallaIngresarDatosActiva;
        }
        if ((ptrContexto->PrevTag >= 60) && (ptrContexto->PrevTag <= 80)) {
                ptrContexto->currIdx = ptrContexto->PrevTag - 60;
                ptrContexto->modo = borrar;
                borrarCoordeanda(ptrContexto);
        }

    }
}

Ft_Gpu_CoCmd_Dlstart(phost);
Ft_App_WrCoCmd_Buffer(phost, CLEAR_COLOR_RGB(32, 32, 32));
Ft_App_WrCoCmd_Buffer(phost, CLEAR(1, 1, 1));

Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(0));

//cuadro de texto
#define cuadrotxt_x1 20
#define cuadrotxt_y1 40
#define cuadrotxt_w1 295
#define cuadrotxt_h1 210

//Dibujar Barra superiot**************************************************************************

Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(64, 64, 64));
```

```
        Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(1 * 16)); //LINE_WIDTH is used for corner curvature
        Ft_App_WrCoCmd_Buffer(phost, BEGIN(RECTS));
        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(0 * 16, 0 * 16));
        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(480 * 16, 20 * 16));

        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0, 0, 0));
        Ft_App_WrCoCmd_Buffer(phost, BEGIN(LINES));
        Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(1 * 16)); //LINE_WIDTH is used for corner curvature
        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(0 * 16, 21 * 16));
        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(480 * 16, 21 * 16));
        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 255));
        Ft_Gpu_CoCmd_Text(phost, FT_DispWidth / 2, 10, 27, OPT_CENTER,
                        "Ingrese coordenadas (tiempo, velocidad)");

        //escribir coordenadas
*********************************************************************
        Ft_App_WrCoCmd_Buffer(phost, SCISSOR_XY(cuadrotxt_x1-10, cuadrotxt_y1-10));
        Ft_App_WrCoCmd_Buffer(phost,
                        SCISSOR_SIZE(cuadrotxt_w1+40, cuadrotxt_h1 + 20));

        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(64, 64, 64));
        Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(10 * 16)); //LINE_WIDTH is used for corner curvature
        Ft_App_WrCoCmd_Buffer(phost, BEGIN(RECTS));
        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(cuadrotxt_x1*16, cuadrotxt_y1*16));
        Ft_App_WrCoCmd_Buffer(phost,
                        VERTEX2F((cuadrotxt_x1+cuadrotxt_w1-30)*16,
                                        (cuadrotxt_y1+cuadrotxt_h1)*16));

        Ft_Gpu_CoCmd_Track(phost, cuadrotxt_x1 + cuadrotxt_w1 - 30, cuadrotxt_y1,
                        50, cuadrotxt_h1, scrollTAG);
        //Ft_Gpu_CoCmd_Track(phost,cuadrotxt_x1,cuadrotxt_y1, cuadrotxt_w1+20,cuadrotxt_h1+200,
scrollTAG);
        static unsigned int scrollval, TrackRegisterVal = 0;

        TrackRegisterVal = Ft_Gpu_Hal_Rd32(phost, REG_TRACKER);

        ft_uint8_t tagval = 0;
        tagval = TrackRegisterVal & 0xff;
        if (0 != tagval) {
                if (scrollTAG == tagval) {
                        scrollval = TrackRegisterVal >> 16;
                        if ((scrollval + 65535 / 10) > (9 * 65535 / 10)) {
                                scrollval = (8 * 65535 / 10);
                        } else if (scrollval < (1 * 65535 / 10)) {
                                scrollval = 0;
                        } else {
                                scrollval -= (1 * 65535 / 10);
                        }
                }
        }
        Ft_Gpu_CoCmd_FgColor(phost, 0x101010);
        Ft_Gpu_CoCmd_BgColor(phost, 0x999999);
        Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(1));
        Ft_App_WrCoCmd_Buffer(phost, TAG(scrollTAG));
        Ft_Gpu_CoCmd_Scrollbar(phost, cuadrotxt_x1 + cuadrotxt_w1 - 20,
                        cuadrotxt_y1 + 5, 30, cuadrotxt_h1 - 10, 0, scrollval,
                        (65535 * 0.2), 65535);
        Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(0));

        Ft_Gpu_CoCmd_Track(phost, cuadrotxt_x1 + cuadrotxt_w1, cuadrotxt_y1, 40,
                        cuadrotxt_h1, scrollTAG);

        Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(5 * 16)); //LINE_WIDTH is used for corner curvature

#define scrollpixeles 900
#define scrolloffset  (int)scrollval*scrollpixeles/65535
        int j = 0;
        //    while( (j < ( ptrContexto->nroCoordenadas )) ){
        for (j = scrolloffset / 40;
```

```
                (j < (ptrContexto->nroCoordenadas))
                                && (j < (scrolloffset + 240) / 30); j++) {
            Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(0));

            Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(240, 240, 240));
            Ft_App_WrCoCmd_Buffer(phost, BEGIN(RECTS));
            Ft_App_WrCoCmd_Buffer(phost,
                            VERTEX2F((cuadrotxt_x1+5)*16,
                                            (cuadrotxt_y1 + (j * 40) - scrolloffset)*16));
            Ft_App_WrCoCmd_Buffer(phost,
                            VERTEX2F((cuadrotxt_x1+cuadrotxt_w1-35)*16,
                                            (cuadrotxt_y1 + (j * 40) - scrolloffset + 25)*16));

            StringArray[0] = '\0';
            strcat(StringArray, "(");
            strcat(StringArray, ptrContexto->vec_Coordenadas[j].time_Array);
            strcat(StringArray, ", ");
            strcat(StringArray, ptrContexto->vec_Coordenadas[j].RPM_Array);
            strcat(StringArray, ")");

            Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0x0, 0x0, 0x0));
            Ft_Gpu_CoCmd_Text(phost, cuadrotxt_x1 + 5,
                            cuadrotxt_y1 + (j * 40) - scrolloffset + 15, 27, OPT_CENTERY,
                            StringArray); //text info

            Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(1));

            Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0xff, 0xff, 0xff));
            Ft_Gpu_CoCmd_FgColor(phost, 0x404080);
            Ft_Gpu_CoCmd_GradColor(phost, 0x00ff00);

            Ft_App_WrCoCmd_Buffer(phost, TAG(j + 20));
            Ft_Gpu_CoCmd_Button(phost, cuadrotxt_x1 + 130,
                            cuadrotxt_y1 + (j * 40) - scrolloffset, 60, 25, 26,
                            tagoptionvec[j], "Editar");
            Ft_App_WrCoCmd_Buffer(phost, TAG(j + 60));
            Ft_Gpu_CoCmd_Button(phost, cuadrotxt_x1 + 200,
                            cuadrotxt_y1 + (j * 40) - scrolloffset, 60, 25, 26,
                            tagoptionvec[j + 20], "borrar");

            //        j++;

        }

        Ft_App_WrCoCmd_Buffer(phost, SCISSOR_XY(0, 0));
        Ft_App_WrCoCmd_Buffer(phost, SCISSOR_SIZE(480, 272));

        //*****************************************************************************************
***
        Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(1));

        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0xff, 0xff, 0xff));
        Ft_Gpu_CoCmd_FgColor(phost, 0x404080);
        Ft_Gpu_CoCmd_GradColor(phost, 0x00ff00);

        Ft_Gpu_CoCmd_FgColor(phost, 0x009900);
        Ft_App_WrCoCmd_Buffer(phost, TAG(boton101));
        Ft_Gpu_CoCmd_Button(phost, 350, 30, 120, 70, 29, tagoptionBtn101,
                        "Confirmar");

        Ft_Gpu_CoCmd_FgColor(phost, 0x404080);
        Ft_App_WrCoCmd_Buffer(phost, TAG(boton102));
        Ft_Gpu_CoCmd_Button(phost, 350, 110, 120, 70, 29, tagoptionBtn102,
                        "Agregar");

        //Ft_Gpu_CoCmd_FgColor(phost,0x404080);
        Ft_App_WrCoCmd_Buffer(phost, TAG(boton100));
        Ft_Gpu_CoCmd_Button(phost, 350, 190, 120, 70, 29, tagoptionBtn100,
                        "Volver");
```

```
        if (flagMsg == 1) {

                Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 255));
                Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(10 * 16)); //LINE_WIDTH is used for corner
curvature
                Ft_App_WrCoCmd_Buffer(phost, BEGIN(RECTS));
                Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(50 * 16, 50 * 16));
                Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(430 * 16, 222 * 16));

                Ft_Gpu_CoCmd_FgColor(phost, 0x404080);
                Ft_App_WrCoCmd_Buffer(phost, TAG(boton104));
                Ft_Gpu_CoCmd_Button(phost, 180, 150, 120, 70, 29, tagoptionBtn104,
                                "Aceptar");

                strcpy(StringArray, "Error!");
                Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0x0, 0x0, 0x0));
                Ft_Gpu_CoCmd_Text(phost, 240, 100, 26, OPT_CENTER, StringArray); //text info
                Ft_Gpu_CoCmd_Text(phost, 240, 130, 26, OPT_CENTER, stringMsg); //text info

        }

        Ft_App_WrCoCmd_Buffer(phost, DISPLAY());
        Ft_Gpu_CoCmd_Swap(phost);

        /* Download the commands into fifo */
        Ft_App_Flush_Co_Buffer(phost);

        /* Wait till coprocessor completes the operation */
        Ft_Gpu_Hal_WaitCmdfifo_empty(phost);

        ptrContexto->PrevTag = CurrTag;

        startHMI = 1;

}

static ft_int32_t ox;
static ft_uint8_t noofch = 0;
ft_int16_t qsin(ft_uint16_t a) {
        ft_uint8_t f;
        ft_int16_t s0, s1;

        if (a & 32768)
                return -qsin(a & 32767);
        if (a & 16384)
                a = 32768 - a;
        f = a & 127;
        s0 = ft_pgm_read_word(sintab + (a >> 7));
        s1 = ft_pgm_read_word(sintab + (a >> 7) + 1);
        return (s0 + ((ft_int32_t) f * (s1 - s0) >> 7));
}

ft_int16_t qcos(ft_uint16_t a) {
        return (qsin(a + 16384));
}

static void polarxy(ft_int32_t r, ft_uint16_t th, ft_int32_t *x, ft_int32_t *y) {
        *x = (16 * (FT_DispWidth / 3) + (((long) r * qsin(th)) >> 11) + 16 * ox);
        *y = (16 * 300 - (((long) r * qcos(th)) >> 11));
}

void vertex(ft_int32_t x, ft_int32_t y) {
        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(x, y));
}

static void polar(ft_int32_t r, ft_uint16_t th) {
        ft_int32_t x, y;
        polarxy(r, th, &x, &y);
```

```
        vertex(x, y);
}

ft_uint16_t da(ft_int32_t i) {
        return (i - 60) * 32768L / 360;
}
static void cs(ft_uint8_t i) {
        switch (i) {
        case 0:
                Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(200, 255, 200));
                break;
        case 90:
                Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 0));
                break;
        case 100:
                Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 0, 0));
                break;
        }
}

#define min(x, y) (((x) < (y)) ? (x) : (y))

static int flagOptimizarPantallaProceso = 0;
void pantallaProceso(contextoPantalla *ptrContexto) {

#define boton120 120
#define boton121 121
#define boton122 122
        static ft_uint8_t CurrTag = 0, Pendown = 1;
        static ft_int32_t tagoptionBtn120, tagoptionBtn121, tagoptionBtn122 = 0;
        ft_char8_t StringArray[20];

        static ft_uint16_t w, h, a, th;
        static ft_int32_t rval, tgt;
        static ft_uint16_t dloffset;
        static ft_uint8_t i, bi, z;
        static ft_int32_t y, tx, ty, o, dt;
        static ft_int16_t val = 0;

//****OPTIMIZACION*********************************************************************************
**********
        if (!flagOptimizarPantallaProceso) {

                flagOptimizarPantallaProceso = 1;

                w = 460.0;
                h = 120.0;
                noofch = 1;
                dt = 10;

                Ft_Gpu_CoCmd_Dlstart(phost);
                Ft_App_WrCoCmd_Buffer(phost, CLEAR_COLOR_RGB(32, 32, 32));
                Ft_App_WrCoCmd_Buffer(phost, CLEAR(1, 1, 1));

                Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(0));

                //Barra sup
                Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(64, 64, 64));
                Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(1 * 16)); //LINE_WIDTH is used for corner
curvature
                Ft_App_WrCoCmd_Buffer(phost, BEGIN(RECTS));
                Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(0 * 16, 0 * 16));
                Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(480 * 16, 20 * 16));

                Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0, 0, 0));
                Ft_App_WrCoCmd_Buffer(phost, BEGIN(LINES));
                Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(1 * 16)); //LINE_WIDTH is used for corner
curvature
                Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(0 * 16, 21 * 16));
```

```
                Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(480 * 16, 21 * 16));
                Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 255));
                Ft_Gpu_CoCmd_Text(phost, FT_DispWidth / 2, 10, 27, OPT_CENTER,
                                "Iniciar proceso");

                Ft_App_WrCoCmd_Buffer(phost, CLEAR_COLOR_RGB(32, 32, 32));
//                      Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));

                y = 25;
                z = 0;
                ox = 240 * z;
                Ft_App_WrCoCmd_Buffer(phost, SCISSOR_XY(ox + dt, y));
                Ft_App_WrCoCmd_Buffer(phost, SCISSOR_SIZE(w, h + 30));
                Ft_App_WrCoCmd_Buffer(phost, CLEAR(1, 1, 1));
                Ft_App_WrCoCmd_Buffer(phost, BEGIN(LINES));
                Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(10));
                for (bi = 0; bi < 111; bi += 10) {
                        cs(bi);
                        for (i = 2; i < 10; i += 2) {
                                a = da(bi + i);
                                polar(220, a);
                                polar(240, a);
                        }
                }

                Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(16));
                for (i = 0; i < 121; i += 10) {
                        cs(i);
                        a = da(i);
                        polar(220, a);
                        polar(250, a);
                }
                Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 255));
                for (i = 0; i < 121; i += 10) {
                        a = da(i);
                        polarxy(260, a, &tx, &ty);
                        Ft_Gpu_CoCmd_Number(phost, tx >> 4, ty >> 4, 26, OPT_CENTER,
                                        i / 10);
                }
                //         ox = (FT_DispWidth/(2*noofch))+(z*(FT_DispWidth/2));
                Ft_Gpu_CoCmd_Text(phost, FT_DispWidth / 3, h + 10, 28, OPT_CENTERX,
                                "RPM x1000");

                Ft_App_Flush_Co_Buffer(phost);
                Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
                dloffset = Ft_Gpu_Hal_Rd16(phost, REG_CMD_DL);

                Ft_Gpu_Hal_WrCmd32(phost, CMD_MEMCPY);
                Ft_Gpu_Hal_WrCmd32(phost, 100000L);
                Ft_Gpu_Hal_WrCmd32(phost, RAM_DL);
                Ft_Gpu_Hal_WrCmd32(phost, dloffset);
                y = 10 + 120 + 20;
                rval = 0, tgt = 4500;

        }
//*************************************************************************************************

        /* Check the user input and then add the characters into array */
        CurrTag = Ft_Gpu_Hal_Rd8(phost, REG_TOUCH_TAG);

        Pendown = ((Ft_Gpu_Hal_Rd32(phost, REG_TOUCH_DIRECT_XY) >> 31) & 0x01);

        if (1 == Pendown) {
                tagoptionBtn120 = 0;
                tagoptionBtn121 = 0;
                tagoptionBtn122 = 0;
        }
        if (Pendown == 0) {
                switch (CurrTag) {
```

```
                case boton120:
                        tagoptionBtn120 = OPT_FLAT;
                        break;
                case boton121:
                        tagoptionBtn121 = OPT_FLAT;
                        break;
                case boton122:
                        tagoptionBtn122 = OPT_FLAT;
                        break;
                }

        }

        if ((CurrTag != ptrContexto->PrevTag)) {
                /* check whether pwndown has happened */
                if ((1 == Pendown)) {
                        switch (ptrContexto->PrevTag) {

                        case boton120:

                                break;
                        case boton121:
                                ;

                                if (flagEstado == 1) {
                                        flagEstado = 0;
                                        //stop
                                        flagDetener = 1;
                                        errorsound();

                                } else if (flagEstado == 0) {
                                        //inicia

                                        buttonsound();

                                        flagEstado = 1;
                                        cargarRef();

                                }

                                break;

                        case boton122:
                                if (flagEstado == 1) {
                                        errorsound();
                                        goto sale;
                                }
                                flagOptimizarPantallaProceso = 1;
                                buttonsound();

                                selectPantalla = pantallaGraficoActiva;
                                return;
                                break;

                        }

                }
        }

        /****GAUGE*************************************************************************************
*********************************/
        sale: Ft_Gpu_CoCmd_Dlstart(phost);
        Ft_Gpu_CoCmd_Append(phost, 100000L, dloffset);
        ox = 0;

        //float feedback_RPM=0;
        //feedback_RPM = ((float)24000000/(float)capI);
        //feedback_RPM = feedback_RPM * 60.0;
        val = (int) wmeter;
```

```
        val = min(12000, val);
        if (flagEstado == 0)
                val = 0;

        Ft_App_WrCoCmd_Buffer(phost, SCISSOR_XY(ox + dt, 10));
        Ft_App_WrCoCmd_Buffer(phost, SCISSOR_SIZE(w, 120));
        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 255));
        Ft_App_WrCoCmd_Buffer(phost, BEGIN(LINES));
        Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(10));
        th = ((ft_uint16_t) val - 6000L) * 32768L / 36000L;
        for (o = -5; o < 6; o++) {
                polar(170, th + (o << 5));
                polar(235, th);
        }

        //**************************************************************************************************
**********************************

        //Alarma**********************************************************************************************
***
        if ((currTime >= (ptrContexto->t_max - 50))
                        && (ptrContexto->nroCoordenadas >= 3)) {
                Alarma();
        }

        //progresbar******************************************************************************************
**

        Ft_App_WrCoCmd_Buffer(phost, SCISSOR_XY(0, 0));
        Ft_App_WrCoCmd_Buffer(phost, SCISSOR_SIZE(480, 272));
        //Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));

        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(20, 20, 20));

        Ft_Gpu_CoCmd_FgColor(phost, 0x0000ff);
        //  Ft_Gpu_CoCmd_GradColor(phost,0x0000ff);
        Ft_Gpu_CoCmd_BgColor(phost, 0x909090);
        Ft_Gpu_CoCmd_Progress(phost, FT_DispWidth / 3 - 180 / 2, 180, 180, 30, 0,
                        currTime / 100, (ptrContexto->t_max) / 100);

        //**************************************************************************************************
***

        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 255));

        sprintf(StringArray, "%d seg / %d seg", (currTime / 1000),
                        (ptrContexto->t_max / 1000));
        Ft_Gpu_CoCmd_Text(phost, FT_DispWidth / 3, 185, 28, OPT_CENTERX,
                        StringArray);     //text info

        //**************************************************************************************************
***

        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0xff, 0xff, 0xff));

        Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(1));

        Ft_Gpu_CoCmd_GradColor(phost, 0x00ff00);

        Ft_Gpu_CoCmd_FgColor(phost, 0x009900);
        Ft_App_WrCoCmd_Buffer(phost, TAG(boton121));

        if (flagEstado == 0) {
                Ft_Gpu_CoCmd_FgColor(phost, 0x009900);
                Ft_Gpu_CoCmd_Button(phost, 390, 140, 80, 70, 27, tagoptionBtn121,
                                "iniciar");

        } else if (flagEstado == 1) {
                Ft_Gpu_CoCmd_FgColor(phost, 0xBA0000);
```

```
                Ft_Gpu_CoCmd_Button(phost, 390, 140, 80, 70, 27, tagoptionBtn121,
                                "Detener");

        }

        Ft_App_WrCoCmd_Buffer(phost, SCISSOR_XY(0, 0));
        Ft_App_WrCoCmd_Buffer(phost, SCISSOR_SIZE(480, 272));
        Ft_Gpu_CoCmd_FgColor(phost, 0x404080);
        Ft_App_WrCoCmd_Buffer(phost, TAG(boton122));
        Ft_Gpu_CoCmd_Button(phost, 390, 50, 80, 70, 27, tagoptionBtn122, "Volver");

        Ft_App_WrCoCmd_Buffer(phost, DISPLAY());
        Ft_Gpu_CoCmd_Swap(phost);
        /* Download the commands into fifo */
        Ft_App_Flush_Co_Buffer(phost);
        /* Wait till coprocessor completes the operation */
        //    Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
        ptrContexto->PrevTag = CurrTag;

}

void pantallaInicio(contextoPantalla *ptrContexto) {

#define boton5 5
#define boton6 6
#define boton7 7

        static ft_int16_t yOffset, xOffset;
        static ft_uint8_t CurrTag = 0, Pendown = 1;
        static ft_int32_t tagoptionBtn5, tagoptionBtn6, tagoptionBtn7 = 0;

        /* Check the user input and then add the characters into array */
        CurrTag = Ft_Gpu_Hal_Rd8(phost, REG_TOUCH_TAG);
        Pendown = ((Ft_Gpu_Hal_Rd32(phost, REG_TOUCH_DIRECT_XY) >> 31) & 0x01);

        /* check whether pwndown has happened */
        if ((CurrTag != ptrContexto->PrevTag)) {
                if (Pendown == 0) {
                        tagoptionBtn5 = OPT_FLAT;
                        tagoptionBtn6 = OPT_FLAT;
                        tagoptionBtn7 = OPT_FLAT;
                }

                if (Pendown == 1) {
                        switch (ptrContexto->PrevTag) {
                        case boton5: //nuevo
                                tagoptionBtn5 = 0;
                                buttonsound();

                                ptrContexto->nroCoordenadas = 0;
                                ptrContexto->flagborrarDisp = false;
                                ptrContexto->currIdx = 0;
                                ptrContexto->t_max = 0;
                                ptrContexto->RPM_max = 0;

                                int j = 0;
                                while (j < 30) {

                                        ptrContexto->vec_Coordenadas[j].RPM_Array[0] = '\0';
                                        ptrContexto->vec_Coordenadas[j].time_Array[0] = '\0';
                                        ptrContexto->vec_Coordenadas[j].time_int = 0;
                                        ptrContexto->vec_Coordenadas[j].RPM_int = 0;
                                        j++;

                                }

                                selectPantalla = pantallaMenuPrincipalActiva;
                                return;
                                break;
```

```
                        case boton6:
                                tagoptionBtn6 = 0;
                                break;
                        case boton7:
                                tagoptionBtn7 = 0;
                                break;
                        }
                }
        }

        Ft_Gpu_CoCmd_Dlstart(phost);
        Ft_App_WrCoCmd_Buffer(phost, CLEAR_COLOR_RGB(32, 32, 32));
        Ft_App_WrCoCmd_Buffer(phost, CLEAR(1, 1, 1));

        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0x00, 0x00, 0x00));
        Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(0));

        //cuadro de texto
#define Boton_w 200
#define Boton_h 100
#define Boton_yOffset 20

        //Barra sup
        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(64, 64, 64));
        Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(1 * 16)); //LINE_WIDTH is used for corner curvature
        Ft_App_WrCoCmd_Buffer(phost, BEGIN(RECTS));
        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(0 * 16, 0 * 16));
        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(480 * 16, 20 * 16));

        Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(10 * 16));
        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(50 * 16, 75 * 16));
        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(430 * 16, 225 * 16));

        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0, 0, 0));
        Ft_App_WrCoCmd_Buffer(phost, BEGIN(LINES));
        Ft_App_WrCoCmd_Buffer(phost, LINE_WIDTH(1 * 16)); //LINE_WIDTH is used for corner curvature
        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(0 * 16, 21 * 16));
        Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(480 * 16, 21 * 16));
        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 255));
        Ft_Gpu_CoCmd_Text(phost, FT_DispWidth / 2, 10, 27, OPT_CENTER, "Inicio");

        Ft_App_WrCoCmd_Buffer(phost, TAG_MASK(1));

        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(0xff, 0xff, 0xff));
        Ft_Gpu_CoCmd_FgColor(phost, 0x404080);
        Ft_Gpu_CoCmd_GradColor(phost, 0x00ff00);

        Ft_App_WrCoCmd_Buffer(phost, TAG(boton5));
        Ft_Gpu_CoCmd_Button(phost, FT_DispWidth / 2 - Boton_w / 2, 100, Boton_w,
                        Boton_h, 30, tagoptionBtn5, "Nuevo");

        /*    Ft_Gpu_CoCmd_FgColor(phost,0x009900);
         Ft_App_WrCoCmd_Buffer(phost,TAG(13));
         Ft_Gpu_CoCmd_Button(phost,cuadrotxt_x - 50 ,cuadrotxt_y + cuadrotxt_h + 100
,100,30,29,tagoption13,"Aceptar");

         Ft_Gpu_CoCmd_FgColor(phost,0xBA0000);
         Ft_App_WrCoCmd_Buffer(phost,TAG(14));
         Ft_Gpu_CoCmd_Button(phost,cuadrotxt_x + cuadrotxt_w - 50,cuadrotxt_y + cuadrotxt_h + 100
,100,30,29,tagoption14,"Cancelar");
         */

        Ft_App_WrCoCmd_Buffer(phost, DISPLAY());
        Ft_Gpu_CoCmd_Swap(phost);

        /* Download the commands into fifo */
        Ft_App_Flush_Co_Buffer(phost);

        /* Wait till coprocessor completes the operation */
```

```
        Ft_Gpu_Hal_WaitCmdfifo_empty(phost);

        ptrContexto->PrevTag = CurrTag;

}

void Alarma(void) {

        Ft_Gpu_Hal_Wr8(phost, REG_VOL_SOUND, 250);
        Ft_Gpu_Hal_Wr16(phost, REG_SOUND, (0xE41E));
        Ft_Gpu_Hal_Wr8(phost, REG_PLAY, 1);
}
void inicio(void) {
        Ft_Gpu_CoCmd_Dlstart(phost);
        Ft_App_WrCoCmd_Buffer(phost, CLEAR_COLOR_RGB(32, 32, 32));
        Ft_App_WrCoCmd_Buffer(phost, CLEAR(1, 1, 1));

        Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 255));
        Ft_Gpu_CoCmd_Text(phost, FT_DispWidth / 2, 50, 30, OPT_CENTER,
                        "Iniciando..."); //text info

        Ft_Gpu_CoCmd_Spinner(phost, 220, 150, 0, 1);

        Ft_App_WrCoCmd_Buffer(phost, DISPLAY());
        Ft_Gpu_CoCmd_Swap(phost);
        Ft_App_Flush_Co_Buffer(phost);
        Ft_Gpu_Hal_WaitCmdfifo_empty(phost);

        delay(2000);

}

void powerdown(void) {
        mute();
        Chip_GPIO_WriteDirBit(LPC_GPIO, LPC_FT800_PD_N_PORT, LPC_FT800_PD_N_BIT,
                        false);

}

ft_void_t setup() {

        Ft_Gpu_HalInit_t halinit;

        Ft_Gpu_Hal_Init(&halinit);

        Ft_Gpu_Hal_Open(&host);

        Chip_SPI_SetBitRate(LPC_SPI, 4000000); //in KHz

        phost = &host;

        SAMAPP_BootupConfig();

//Chip_SPI_SetBitRate(LPC_SPI, 2000); //in KHz
        /*It is optional to clear the screen here*/
        Ft_Gpu_Hal_WrMem(phost, RAM_DL, (ft_uint8_t *) FT_DLCODE_BOOTUP,
                        sizeof(FT_DLCODE_BOOTUP));
        Ft_Gpu_Hal_Wr8(phost, REG_DLSWAP, DLSWAP_FRAME);

        Ft_Gpu_Hal_Sleep(1000); //Show the booting up screen.

        /*interrupt enbale*/
        Ft_Gpu_Hal_Wr8(phost, REG_INT_MASK, INT_CONVCOMPLETE); //INT_TAG);
        Ft_Gpu_Hal_Wr8(phost, REG_INT_EN, 0x01);

        ft_delay(100);

        Ft_Gpu_Hal_Rd8(phost, REG_INT_FLAGS);
```

```
        /* Sample code for GPU primitives */
//SAMAPP_CoPro_Widget_Calibrate();

        Ft_App_WrDlCmd_Buffer(phost, CLEAR(1, 1, 1)); // clear screen
        Ft_App_WrDlCmd_Buffer(phost, COLOR_RGB(255, 0, 0));
        Ft_App_WrDlCmd_Buffer(phost, STENCIL_OP(INCR, INCR));
        Ft_App_WrDlCmd_Buffer(phost, COLOR_MASK(0, 0, 0, 0)); //mask all the colors
        Ft_App_WrDlCmd_Buffer(phost, BEGIN(EDGE_STRIP_L));
        Ft_App_WrDlCmd_Buffer(phost,
                        VERTEX2II((FT_DispWidth / 2), (FT_DispHeight / 4), 0, 0));
        Ft_App_WrDlCmd_Buffer(phost,
                        VERTEX2II((FT_DispWidth * 4 / 5), (FT_DispHeight * 4 / 5), 0, 0));
        Ft_App_WrDlCmd_Buffer(phost,
                        VERTEX2II((FT_DispWidth / 4), (FT_DispHeight / 2), 0, 0));
        Ft_App_WrDlCmd_Buffer(phost,
                        VERTEX2II((FT_DispWidth / 2), (FT_DispHeight / 4), 0, 0));
        Ft_App_WrDlCmd_Buffer(phost, END());
        Ft_App_WrDlCmd_Buffer(phost, COLOR_MASK(1, 1, 1, 1)); //enable all the colors
        Ft_App_WrDlCmd_Buffer(phost, STENCIL_FUNC(EQUAL, 1, 255));
        Ft_App_WrDlCmd_Buffer(phost, BEGIN(EDGE_STRIP_L));
        Ft_App_WrDlCmd_Buffer(phost, VERTEX2F(FT_DispWidth * 16, 0));
        Ft_App_WrDlCmd_Buffer(phost,
                        VERTEX2F(FT_DispWidth * 16, FT_DispHeight * 16));

//Ft_Gpu_Hal_Wr8(phost,REG_PWM_DUTY,100);

        Ft_App_WrDlCmd_Buffer(phost, END());

}
```

Main.c

```
#include "FT_Platform.h"
#include "board.h"
#include "controlador.h"
#include "PWM.h"
#include "driver.h"
#include  "inputCAP.h"
#include  "adc.h"
#include  "global.h"
#include  "ritimer.h"
#include  "inrush_mos.h"

volatile int pausemscont = 0;
volatile uint32_t capI = 0;

void SysTick_Handler(void) //10ms
{

        static uint32_t time_10ms, count10ms, count100ms = 0, count1s = 0;
        static contextoPantalla contexto;

        count10ms++;
        count100ms++;
        count1s++;
        if (pausemscont)
                pausemscont--;

        if (count10ms == 2) {

                if (startHMI) {
                        switch (selectPantalla) {
                        case pantallaIngresarDatosActiva:
                                pantallaIngresarDatos(&contexto);
                                break;
```

```
                        case pantallaMenuPrincipalActiva:
                                pantallaMenuPrincipal(&contexto);
                                break;
                        case pantallaInicioActiva:
                                pantallaInicio(&contexto);
                                break;
                        case pantallaGraficoActiva:
                                pantallaGrafico(&contexto);
                                break;
                        case pantallaProcesoActiva:
                                pantallaProceso(&contexto);
                                break;

                        }

                }
                count10ms = 0;
        }
        if (count100ms == 10) {
                count100ms = 0;

        }

        if (count1s == 1000) {
                count1s = 0;

        }

        time_10ms++;

}

void TIMER3_IRQHandler(void) {
        static uint32_t a, b, j = 0;
//snesor corriente
        if (Chip_TIMER_CapturePending(LPC_TIMER3, 0)) {
                Chip_TIMER_ClearCapture(LPC_TIMER3, 0);
                if (j == 0) {
                        a = Chip_TIMER_ReadCapture(LPC_TIMER3, 0);
                }
                if (j == 12) {
                        b = Chip_TIMER_ReadCapture(LPC_TIMER3, 0);

                        capI = b - a;

                        a = b;
                        j = 0;
                }

                j++;

        }

}

void delay(int ms) {
        pausemscont = ms;
        while (pausemscont)
                ;
}

int main(void) {

        /*vercion
         /* Initialize board and chip */

        SystemCoreClockUpdate();
        Board_Init();
```

```
        startHMI = 0;
        SysTick_Config(SystemCoreClock / 1000);

        //inicilaizacion pantalla
        setup();
        inicio();

        //inicilacizacion perifericos
        Inrush_mosINI();
        ADCINI();
        PWMINI();
        CAPINI();
        driverINI();
        sd_off();
        RITINI();

        tensionMOTOR(0);
        CaptureEnable();

        SysTick_Config(SystemCoreClock / 100);
        startHMI = 1;
        selectPantalla = pantallaInicioActiva;
        Inrush_mosON();

        while (1) {

                /*      float Va;
                 if (readADC(ADC_Vbus_CHANNEL)) Va = readADC(ADC_Vbus_CHANNEL)/65.06;
                 if ((Va < 43.0) && Va ){
                 powerdown();
                 }
                 */

        }
}
```

ANEXO II –Esquematicos