UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SISTEMA DE MANEJO DE ENERGÍA Y MONITOREO DE CENTRAL MICRO HIDRÁULICA PLUG & PLAY

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

RENÉ ANGELLO ROSATI PÉREZ

PROFESOR GUÍA:
RODRIGO PALMA BEHNKE

MIEMBROS DE LA COMISIÓN:
GUILLERMO JIMÉNEZ ESTEVEZ.
ARIEL VALDENEGRO ESPINOZA.

SANTIAGO DE CHILE
2015

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO
POR: RENÉ ANGELLO ROSATI PÉREZ
PROF. GUÍA: Dr. RODRIGO PALMA BEHNKE
FECHA: ENERO 2015

SISTEMA DE MANEJO DE ENERGÍA Y MONITOREO DE CENTRAL MICRO HIDRÁULICA PLUG & PLAY

La matriz energética del Sistema Interconectado Central (SIC) está compuesta prioritariamente por medios de generación con base a combustibles fósiles y grandes centrales de generación hidráulica. El Gobierno de Chile decidió impulsar el desarrollo de proyectos de generación en base a Energías Renovables No Convencionales (ERNC) [1]. Este desafío incorpora el desarrollo de proyectos hidráulicos de pequeñas unidades hidráulicas de pasadas para generar energía, donde el país posee un gran potencial.

En particular, en el Centro de Energía de la Universidad de Chile (CE-FCFM) se está desarrollando una central Micro Hidráulica de 10 kW, la que tiene la capacidad de operar de forma aislada o en sincronismo con la red. Esta central es manejada a través de una Interfaz Hombre Máquina (HMI, *Human Machine Interface*), y se comunica con el controlador principal de la central mediante protocolo serial RS232.

El objetivo de este trabajo es mejorar la HMI existente de modo que presente los datos relevantes del sistema, además de permitir controlar la central de forma sencilla e intuitiva. Esta interfaz debe ser compatible con el esquema de estados y transiciones con el cual opera la central, el que debe ser flexibilizado y ordenado.

Para este trabajo se utiliza el software de diseño de aplicaciones graficas *QTcreator*, que permite de manera sencilla implementar interfaces gráficas, mediante la utilización de objetos predefinidos y código fuente en C++. A su vez, para trabajar en el sistema de estados y transiciones, se utiliza *Code Composer Studio*, el cual permite programar el Procesador Digital de Señales que controla la central.

Se logra crear una interfaz gráfica que cuenta con distintas pestañas en las que se pueden observar las variables relevantes del sistema (tensión, corrientes, potencia, etc.) y enviar órdenes de cambio de estados o modos de operación al controlador principal (Detenido, En Espera, Isla, Sincronizado, Red y Mantencion). Asimismo, se expanden los estados de operación y se reescribe la estructura del programa del controlador principal de forma que quede ordenado de manera secuencial. La solución es integrada y validada exhaustivamente en el prototipo de central micro hidráulica, donde asimismo, una serie de problemas de operación, componentes e interfaz son detectados y abordados sistemáticamente proponiendo soluciones en cada caso.

Como trabajo futuro, se propone la implementación de un sistema de reinicio de la central micro hidráulica en caso de emergencia, como también la implementación de las protecciones contra fallas que puedan ocurrir en el sistema y la subsecuente integración de inteligencia en la

central. También se propone implementar la sección de herramientas de la interfaz, y finalmente la interfaz misma para dispositivos móviles.

Agradecimientos:

Gracias Dios, por todo lo que has hecho.

También debo agradecerle a mi familia. A mi mamá y papá por el apoyo incondicional que me han brindado a lo largo de los años. Por las cosas más básicas que me han enseñado, como la perseverancia y el esfuerzo. A mis hermanos por distraerme o quitarme las distracciones cuando fue necesario. A mis abuelos por el cariño que me han brindado y por incentivar mi creatividad.

A mis profesores por enseñarme la rama de la ingeniería que más me gusta. Por permitirme trabajar en esta memoria y por el apoyo que me han brindado a lo largo de esta carrera. También a mis profesores del colegio por las bases que me entregaron.

Quiero agradecer a Carlos y a Enrique, con los que trabajé en el proyecto de la µHidro en el cual este trabajo se encuentra enmarcado. A mis amigos, tanto de la universidad como del colegio, por todos esos momentos de relajo y diversión que compartimos.

También quería agradecer a Jorge, Matías, y Raúl, por ayudarme con la corrección de este trabajo.

Tabla de Contenido

Índice de Tablas

Índice de Figuras

Acrónimos

CE-FCFM	Centro de Energías de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile
MHPP	Micro Hidráulica Plug & Play
ERNC	Energías Renovables No Convencionales
DSP	Digital Signal Processor
SBC	Single Board Computer
PWM	Pulse Width Modulation
CAN	Controller Area Network
CCS	Code Composer Studio
USB	Universal Serial Port
OTG	On The Go
UART	Universal Asynchronous Receiver Transmitter
NRZ	Non Return to Zero
SCI	Serial Communication Interface
SPI	Serial Peripheral Interface
FIFO	First In First Out
HMI	Human Machine Interface
SDK	Software Development Kit
ADC	Analog to Digital Converter
ADT	Android Development Tools
NDK	Necessary Development Kit

Capítulo 1. Introducción

Actualmente en Chile la matriz energética del Sistema Interconectado Central (SIC) está compuesta prioritariamente por medios de generación con base a combustibles fósiles y grandes centrales de generación hidráulica. Los combustibles fósiles tienen una participación de más del 50% de la capacidad instalada, mientras que la generación hidráulica de un 45% [2]. Estos medios de generación poseen un alto grado de impacto ambiental, ingresando gases de efecto invernadero o inundando grandes áreas de tierra. Con base a esta situación, y sucesos como el terremoto del 2010 con la subsecuente caída del sistema, el Gobierno de Chile decidió impulsar el desarrollo de proyectos de generación basados en ERNC.

En este marco, el Centro de Energía de la Universidad de Chile (CE-FCFM) en conjunto con apoyo de emprendedores del mundo privado decidió emprender un proyecto de generación distribuida con base a energía hidráulica llamado Micro Hidráulica *Plug & Play* (MHPP). Esta unidad tiene como fin aprovechar pequeños recursos hídricos en el sur de Chile, proveyendo al consumidor de una fuente segura y limpia de energía. Este proyecto consta de una micro central hidráulica de pasada, cuya potencia máxima es de 10kW. El objetivo final de este proyecto es el desarrollo de una unidad con proyección comercial que pueda operar tanto de forma aislada o conectada a la red, siendo capaz de detectar cuando esta última se encuentra en un estado de falla para poder aislarse y mantener su suministro local.

De forma adicional, la central cuenta con una interfaz gráfica en la cual se pueden observar las distintas variables del sistema: tensión, potencia generada, estado de operación, etc. Además, ésta permite que el usuario ingrese cambios en la forma de operación de la central. Esta Interfaz Hombre - Máquina (HMI) corresponde a un *Single Board Computer* (SBC), el cual es básicamente un computador con una pantalla táctil. Esta posee la particularidad de contar con una gran cantidad de puertos periféricos con los cuales se puede comunicar con el procesador central.

El computador central de la unidad consta de un Procesador de Señales Digitales (DSP, *Digital Signal Processor*) en el cual se ejecutan todos los algoritmos de control de este sistema, exceptuando el controlador de frecuencia de la central. Este DSP cuenta con una gran variedad de periféricos siendo ideal para este tipo de aplicaciones.

Durante este trabajo se abordó el diseño de la HMI desde el punto de vista de software, lo que incluye la disposición grafica de los elementos que lo componen. Además, se comentan algunas reglas generales que se deben seguir al construir una interfaz gráfica. También se propone e implementa un sistema supervisor de la central encargado de entregar las referencias a los sub controles del sistema, realizar las transiciones de un estado a otro, y detectar fallas en el sistema. Por último, se implementa el sistema de comunicación entre la HMI y el DSP.

1.1. Motivación

La MHPP cuenta con una interfaz gráfica montada sobre una SBC, la cual presenta algunos problemas en su funcionamiento, por lo que surge la necesidad de implementar una nueva versión de la HMI. En particular, posee una pobre respuesta frente a interacciones con el usuario, como retardos en la visualización de información o el congelamiento de la pantalla.. Junto con la actualización de *hardware*, se requiere rediseñar la información desplegada en la interfaz, dado que en la antigua versión algunas de las pestañas presentaban una sobrecarga de información, mientras que otras contenían elementos difíciles de ocupar, tales como barras deslizantes y diales.

Debido a esto, se decide emprender la tarea de realizar una nueva versión de la HMI utilizando el mismo software de diseño utilizado en la versión anterior: *QT Creator*, que permite diseñar de manera gráfica el aspecto de las distintas pestañas de la aplicación. Por otro lado, dado que la HMI debe ser compatible con el software interno del controlador, se abre la posibilidad de modificar el sistema de manejo de energía de la central.

1.2. Objetivos

El objetivo principal de este trabajo de memoria es el diseño e implementación de un sistema de monitoreo de la central micro hidráulica. Sobre la base de un diseño conceptual el sistema resultante debe ser capaz de comunicarse con la central mediante el protocolo serial RS232, y mostrar los distintos estados de operación de la central, variables de estado. Además, debe permitir efectuar cambios de consignas de operación, por lo que resulta obligatorio que la interfaz interactúe con el sistema de control interno de la central. En este último ámbito, debe ser factible expandir el sistema de control supervisor de la central.

1.3. Alcances

Durante este trabajo se abarca la implementación de la HMI sobre la plataforma QT. En ningún momento se aborda la implementación de esta sobre algún otro dispositivo, además de una pequeña prueba para estudiar la factibilidad de la comunicación entre el DSP y una Tablet. Respecto al sistema de control de la central, esta memoria solo se atañe a la implementación del sistema de control supervisor de la central, abarcando la estructura de estados y transiciones de ésta, sin entrar en mayor detalle en la implementación de la inteligencia relacionada con la operación automática, y solo se mencionan y explican superficialmente los controles de tensión, factor de potencia y frecuencia.

1.4. Estructura de Trabajo

En el primer capítulo de este documento, se realiza una breve introducción de los acontecimientos que dieron origen al proyecto que enmarca este trabajo. Además, se exponen los objetivos generales de este tema de memoria, limitaciones y alcances.

En el segundo capítulo, se exponen algunos de los avances ya realizados durante el transcurso del proyecto Micro Hidráulica Plug & Play, tales como la estructura general del sistema, estados básicos de operación, versiones previas de la interfaz gráfica, etc.

En el tercer capítulo, se detallan los requerimientos que deben cumplir el sistema de control y la HMI. Luego, se propone una estructura de estados y transiciones, en la cual se basará el sistema de control supervisor y del sistema de la HMI, junto con la disposición gráfica y la implementación de ambos sistemas. Finalmente, se exponen los criterios de validación de ambos sistemas.

En el cuarto capítulo, se exponen los resultados obtenidos luego de la implementación de ambos sistemas. En particular, se muestran los resultados de los test de validación de ambos sistemas. Luego, se procede a realizar un análisis de estos resultados.

Finalmente, en el quinto capítulo, se exponen las conclusiones de este trabajo, junto con una propuesta de mejoras futuras a realizar en este proyecto.

Capítulo 2. Antecedentes y Marco Teórico

2.1. Micro Hidráulica Plug & Play

2.1.1. Descripción General

La gran cantidad de pequeños recursos hídricos en Chile, junto con los incentivos que ha creado el Gobierno para impulsar la generación distribuida, llevaron al CE-FCFM a emprender un proyecto en el cual se desarrolle una micro central hidráulica de pasada, partiendo desde un equipo de generación hidráulica de origen chino de 10kW. Este equipo sería modificado confiriéndole inteligencia que le permitiría opera de forma aislada o en sincronía con la red. Esta unidad debía de ser competitiva en el mercado, fácil de instalar y compatible con los actuales esquemas de generación distribuida [3].

Para poder llevar a cabo este proyecto se escogió una turbina tipo Turgo (Figura 2-1) que se encuentra acoplada a un generador sincrónico de 10 kVA. El caudal de entrada es regulado por una válvula de mariposa, la cual es actuada por un servo motor.

Figura 2-1: Jet inclinado de una turbina Turgo [3].

Para caracterizar el complejo turbina-generador y probar los distintos prototipos de control, se implementó un pequeño laboratorio en el taller mecánico de Molina, cuya configuración se muestra en la Figura 2-2.

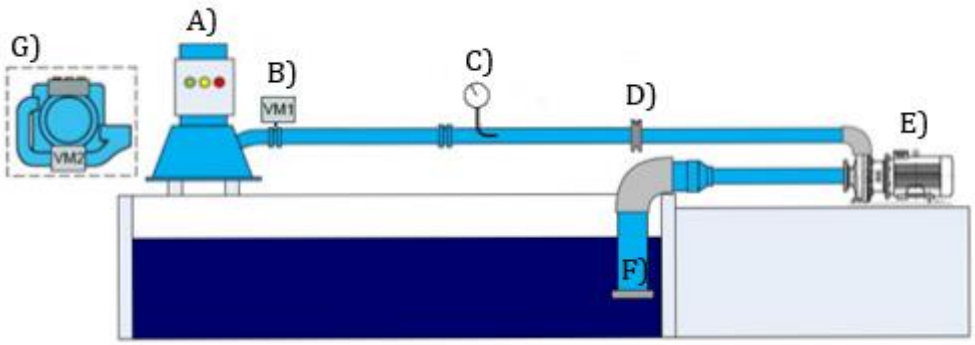


Figura 2-2: Laboratorio de la central micro hidráulica [3].

En la Figura 2-2 se observa la disposición de los equipos en el laboratorio de Molina. Estos equipos son: A) complejo turbina-generador, B) válvula de mariposa, C) Pitor (medidor de presión total), D) placa de orificios (medidor de presión diferencial), E) bomba hidráulica y finalmente F) válvula anti-retorno. En G) se muestra una vista superior del complejo turbina-generador, en la cual se puede observar que hay dos entradas de agua hacia la turbina. Se puede cerrar la entrada trasera de la turbina mediante una segunda válvula de mariposa.

La bomba de agua es controlada por un variador de frecuencia, que se encuentra conectado a la red trifásica del lugar. El caudal generado es medido por unas placas de orificios conectadas a un medidor diferencial de presión. También se cuenta con un medidor de presión total con el cual se puede determinar la altura simulada por el sistema.

El complejo turbina generador se encuentra dentro de una sala de máquina que posee una forma cónica y fue construida utilizando fibra de vidrio. En la parte superior de la casa de máquinas se encuentra la interfaz gráfica, que cuenta con una pantalla táctil (Figura 2-3).

Figura 2-3: Interfaz de usuario [4].

Junto a la turbina existe un tablero de conexiones al cual llegan los cables del generador, la red trifásica, el consumo local y el sistema de control de frecuencia. A un costado de este tablero se encuentran las placas que conforman el prototipo del controlador de la central.

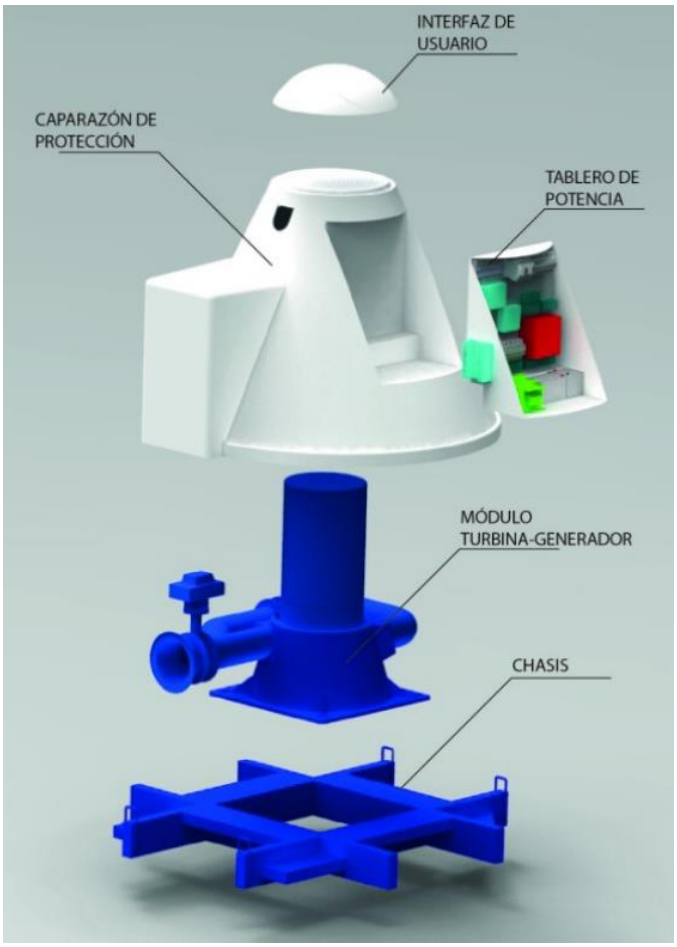


Figura 2-4: Diseño de la "sala" de máquinas de la central [3].

En la Figura 2-4 se muestra la sala de máquinas de la central. En ésta se encuentran el complejo turbina-generador, la HMI, el tablero de potencia y la electrónica de control. Esta última irá colocada en la parte inferior del compartimento en donde se encuentra el tablero de potencia.

2.1.2. Principio de Funcionamiento y Componentes de una Central Hidráulica de Pasada

La generación hidroeléctrica se basa en extraer la energía cinética del agua, a través de un complejo turbina-generador. Al impactar el agua sobre los alabes de la turbina, se provoca que el rotor de ésta comience a girar transformando la energía cinética del agua en energía rotacional. La corriente que circula por el devanado del rotor y el movimiento de este provoca una variación de flujo magnético en los devanados del estator, generando así una tensión inducida en los bornes del generador [5].

La potencia hidráulica de una central hidroeléctrica está determinada por:

$$\boldsymbol{P = \eta\, \delta\, Q\, h} \qquad \textbf{2.1}$$

Donde δ es la densidad del agua (kg/m^3), Q es el caudal (m^3/s), h es la altura del salto del agua (m) y η es la eficiencia de la conversión. Al caer el agua por la tubería de presión, esta gana la energía cinética que hace rotar a la turbina. Esto constituye el principal método ocupado en Chile para extraer energía del agua. El tipo de turbina utilizada en la central depende del caudal del afluente y el tipo de salto. En el caso de Chile, se suelen utilizar turbinas tipo Pelton, las cuales están hechas para poco caudal pero, grandes saltos. A continuación se exponen las principales obras hidráulicas que requiere una central de pasada: [6]

1. **Bocatoma:** la bocatoma es una obra hidráulica cuya función es captar un determinado caudal del afluente que suministrará el agua a la central. Además, debe ser capaz de evitar el ingreso de materiales sólidos y flotantes que se encuentren en el río. Usualmente se emplazan en tramos rectos del rio o el inicio de una curva.

2. **Canal de derivación:** su función es proveer una ruta donde el agua circula desde la bocatoma hasta el desarenador y la cámara de carga. Se deben tratar de evitar las pérdidas por filtración a lo largo de éste.

3. **Desarenador:** es una obra que asemeja un estanque donde la velocidad del agua es reducida para remover las pequeñas partículas de sólidos en suspensión. Estas partículas tienen una naturaleza muy abrasiva, siendo vital que se remuevan antes de que entren a la tubería de presión.

4. **Cámara de carga:** es un estanque de regulación que asegura el correcto funcionamiento de la central, ya que regula el flujo de agua que ingresa a la turbina. Además, puede amortiguar las ondas de presión que se producen cuando la turbina se cierra.

5. **Tubería de presión:** esta tubería se encuentra entre la cámara de carga y la turbina. Debe ser capaz de resistir los cambios de dirección requeridos por las irregularidades del terreno, por lo que debe estar correctamente anclada.

6. **Sala de máquinas:** es una sala en donde se aloja el complejo turbina-generador. Además, también debe resguardar del clima los distintos equipos de control y potencia.

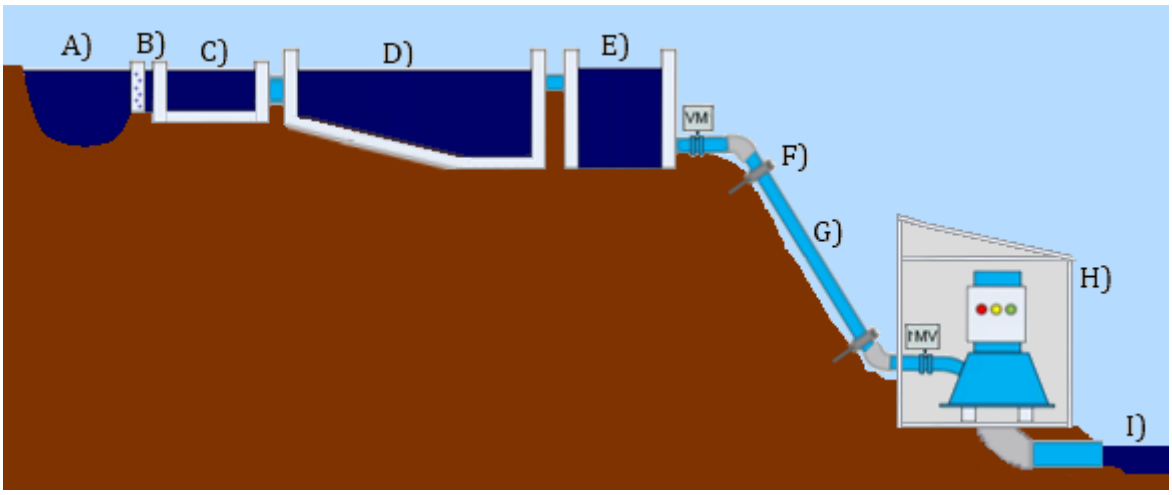


Figura 2-5: Esquema de obras hidráulicas de una central hidroeléctrica de pasada.

En la Figura 2-5 se observa un esquema de la disposición de las distintas obras que se deben realizar en una central de pasada. En este sistema el agua es captada del afluente A) por la bocatoma B), en la cual se eliminan elementos solidos que se encuentran en el afluente. Luego es conducida por el canal de derivación C) hasta el desarenador D), donde se eliminan las partículas en suspensión. Luego, el agua pasa a la cámara de carga E), la cual tiene el objetivo de regular el flujo de agua que circulará por la tubería de presión G), la que se encuentra sujeta por pernos de anclaje F). Finalmente el agua a presión llega hasta la sala de máquinas H) en donde se encuentra la turbina y el generador junto con todos los equipos de control y potencia del sistema. Una vez que el agua pasa por la turbina es liberada en el canal de descarga I).

2.1.3. Sistema Eléctrico

El sistema eléctrico está compuesto por cuatro partes: el generador, la resistencia de desahogo, el consumo local y la red eléctrica. Todas las partes están unidas con una barra mediante el uso de automáticos y en algunos casos mediante contactores. Los contactores son equipos capaces de abrir una conexión bajo carga, y que además pueden ser accionados de forma remota a través de una señal eléctrica.

En un principio se esperaba que una válvula de mariposa capaz de regular el caudal de entrada a la central. Sin embargo, estudios revelaron que esta no era capaz de desempeñar esta función ya que no posee un control fino en el tramo en que se comienza a cerrar, por lo que prácticamente no se cuenta con un control de caudal. Es por esto que resulta necesario consumir la potencia no utilizada por la carga en modo isla, con una resistencia de desahogo, de modo de poder regular la frecuencia del sistema.

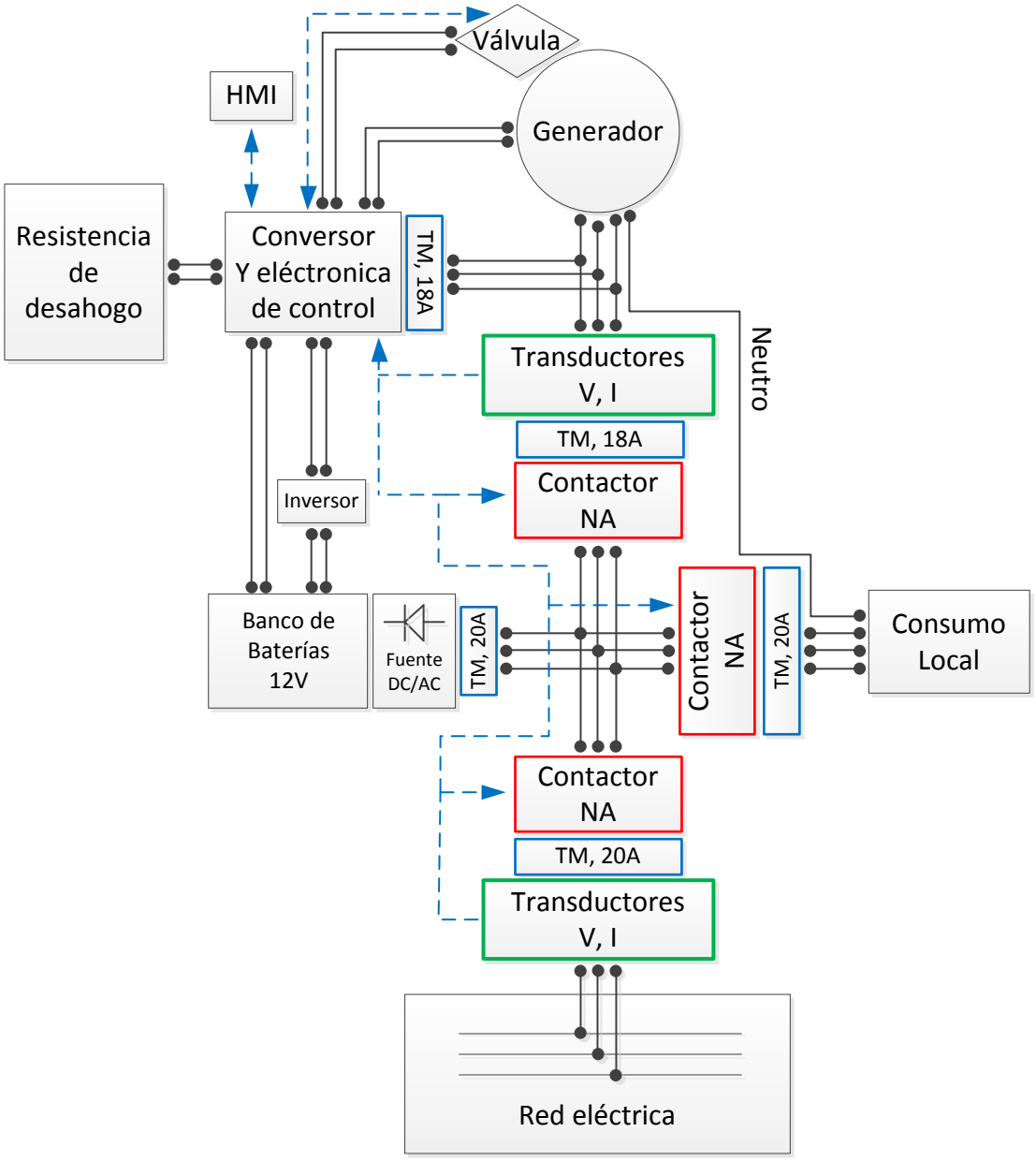


Figura 2-6: Sistema Eléctrico.

El sistema en la Figura 2-6 corresponde al sistema final para ser montado en la sala de máquinas de la central. A la fecha, en el laboratorio de Molina no se está utilizando el sub sistema del banco de baterías e inversor, debido a que para las pruebas en este laboratorio se utiliza la red eléctrica como fuente de alimentación para la electrónica de control y el circuito de campo del generador. Además, la electrónica de control y la de potencia se encuentran separadas. Como se observa en la figura el sistema consta de una barra trifásica principal a la cual se conectan el generador, el consumo local y la red eléctrica mediante un automático y un

contactor. Desde este punto también se conecta la fuente DC que alimenta la electrónica de control y las baterías del sistema. Finalmente en los bornes del generador se conecta el conversor el cual rectifica y consume la energía sobrante del sistema. Otro elemento que se encuentra fuera de servicio en esta etapa corresponde a la válvula de mariposa, debido a que se cuenta con una bomba para simular el caudal.

2.2. Trabajos Previos

En el inicio de este trabajo se habían realizado grandes avances en el proyecto de la central micro hidráulica. En particular, en el área eléctrica, se había desarrollado un prototipo de control en el cual los distintos estados de operación funcionaban correctamente, con excepción del estado de funcionamiento en modo isla. Durante estos trabajos se desarrollaron prototipos de todas las placas electrónicas necesarias y paneles de potencia, además de desarrollar un prototipo de HMI que fue montada sobre una SBC, siendo capaz de controlar la operación de la máquina.

2.3. Estados de Operación y Técnicas de Control

La central Micro hidráulica originalmente poseía cuatro estados de operación: Detenido, En Espera, Aislado y Sincronizado. En la Figura 2-7 se muestra un diagrama que relaciona estos estados.

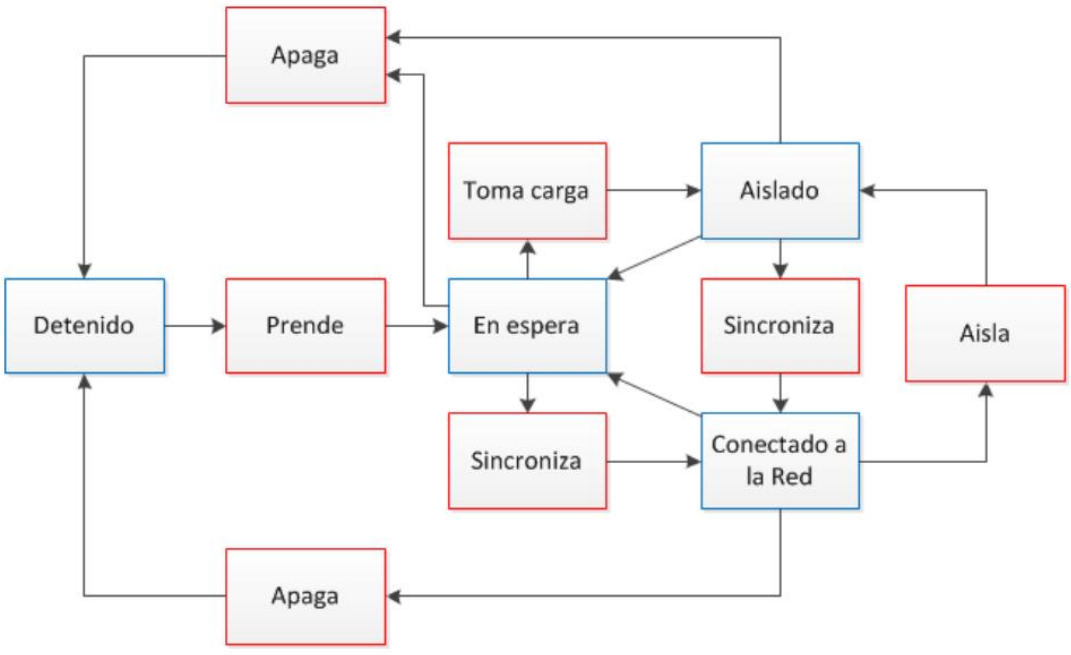


Figura 2-7: Esquema de modos de operación de la Central Micro Hidráulica [4].

En el diagrama presentado en la Figura 2-7 se observan con azul los estados de operación de la central y en rojo las transiciones entre dichos estados. Estas transiciones eran disparadas desde la SBC. La central cuenta con consignas de operación, las cuales se seleccionan en la

interfaz táctil. Estas consignas son transmitidas al DSP, el cual tiene programadas las distintas estrategias de control. Las acciones que toma el DSP dependen de la consigna en la que se encuentra: Prende, Apaga, Automático y Manual.

Las consignas de Automático y Manual controlan la forma en que la central opera. En modo automático la central toma sus propias decisiones de operación tratando de sincronizarse a la red, o generar un sistema aislado. Por otro lado, en la consigna manual el operador, puede escoger en qué modo quiere operar la central, siempre sea factible [4].

De este modo, si la central se encuentra en modo automático, solo basta con enviar la consigna de prender y la central tomará sus propias decisiones de operación, sin intervención del usuario.

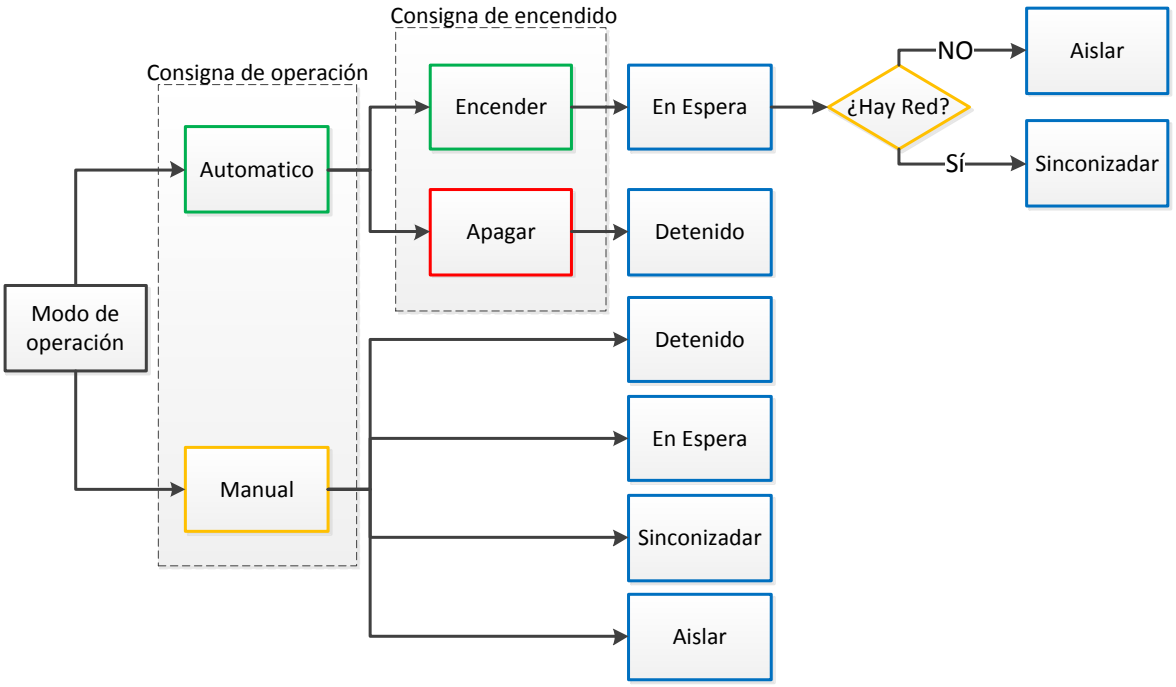


Figura 2-8: Esquema de consignas.

En la Figura 2-8 se muestra el esquema de consignas de la central. Como se mencionó estas se modifican en la HMI. Para esto la HMI se vale de un botón de encendido y apagado, el cual solo funciona cuando la central se encuentra en modo automático. Este botón da la orden a la central de comenzar a operar, y sincronizarse o aislarse de la red según sus condiciones. Por otro lado, si la central está en modo manual, en la interfaz se puede escoger el modo de operación de la central.

En el modo de operación en sincronismo con la red, la central estará inyectando toda la energía generada a la red, la cual le impondrá su propia frecuencia eléctrica y absorberá las variaciones de carga en el consumo local. En cambio, en la operación en isla, la central es la que determina la frecuencia de la red, por lo que la estrategia de control debe compensar las variaciones de carga en el consumo local.

Cuando esta se encuentra Detenida, la central se desconecta de la red y del consumo local, además mantiene la válvula de mariposa cerrada. En este estado se considera que la central se encuentra apagada. Por el contrario, el estado En Espera, la centrar trata de llegar al punto de operación de 220 V, 50 Hz y luego intenta sincronizarse con la red y conectar la carga, o solamente conectar la carga en el caso en que la red no esté disponible [4].

Para mantener una operación estable en alguno de los distintos estados de la máquina se requiere medir y actuar sobre las distintas variables eléctricas del sistema. Para realizarlo, la central cuenta con una unidad de medición, la cual está compuesta por cuatro transductores de tensión y cuatro transductores de corriente. Estos transductores envían los datos medidos mediante señales analógicas hacia el DSP, el cual capta estas señales mediante puertos de Conversión Análoga digital (ADC, *Analog-to-Digital Converter*). Estos datos se obtienen de las fases A y B del generador y de la red.

Estas mediciones son procesadas mediante filtros para obtener una buena estimación de las variables de tensión y corriente del sistema. El siguiente paso es estimar la tensión y la corriente que circula por la fase C del generador y la red. Para esto se supone que el sistema se encuentra en equilibrio. Una vez que se obtienen las estimaciones anteriores se procede a calcular la potencia activa y reactiva que genera la central.

Además de las potencias, el sistema también calcula una estimación para la frecuencia del sistema y el ángulo de fase de la unidad y la red. Esto se realiza mediante un algoritmo llamado *Phase Lock Loop* (PLL). Básicamente, este algoritmo iguala la frecuencia de una señal sinusoidal generada digitalmente, con la señal a la cual se le quiere determinar su frecuencia. En este caso se realizaron los siguientes pasos:

1. Utilizar la transformada de Clark sobre las tensiones fase neutro.
2. Normalizar las tensiones en alfa-beta.
3. Calcular el componte en cuadratura de la tensión.
4. Filtrar una vez para obtener una estimación de la fase (filtro pasa bajo).
5. Filtrar una segunda vez para obtener una estimación de la frecuencia (filtro pasa alto).
6. Normalizar la fase entre -2π y 2π.

Además del sistema de medición, existen sistemas de control encargados de mantener estas variables dentro de los márgenes permitidos. Existen tres sistemas: el sistema de control de frecuencia, el controlador de tensión y finalmente el control de factor de potencia. En dichos sistemas solo la frecuencia es regulada a través de un dispositivo externo.

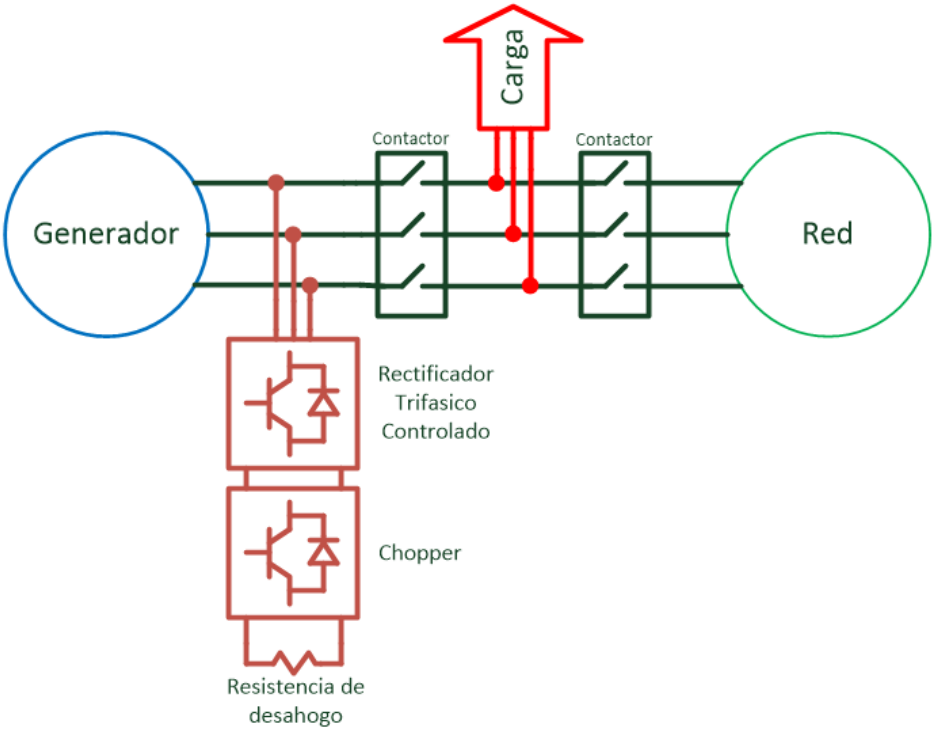


Figura 2-9: Sistema de control de frecuencia.

El sistema consta de un rectificador controlado, un chopper y una resistencia de desahogo, (Figura 2-9). El dispositivo permite deshacerse de la energía sobrante en el sistema de modo que disminuye la frecuencia si sobrepasa los 50 Hz. Dado que este dispositivo no posee la capacidad de inyectar energía al sistema, no puede regular la frecuencia bajo los 50 Hz.

Por otro lado, para controlar la tensión de generación y el factor de potencia de la máquina, se manipula la corriente de campo del generador. La central cuenta con un circuito de chopper que consta de un transistor excitado por un PWM (*Pulse Width Modulation*) en el DSP. De este modo, cuando el ancho de pulso disminuye la tensión disminuye lo que provoca que la corriente también disminuya. Por el contrario, si el ciclo de trabajo aumenta entonces la tensión aumenta y también lo hace la corriente. Este sistema funciona a una tensión de 24 V y posee una corriente máxima de 2A.

El controlador de este sistema depende del estado de operación de la central. Si se encuentra operando en modo sincronizado se ocupara el control por factor de potencia. Por el contrario, si se encuentra en cualquier otro modo se ocupara el control de tensión.

En el caso de control por factor de potencia, se ajusta la corriente de campo en función de φ manteniendo como referencia un factor de potencia 0.95 capacitivo. El control de tensión toma como referencia la tensión de bornes de la máquina tratando de mantenerla en 220V.

Figura 2-10: DSP TMS320F28335 Ralease 2.2.

Los sistemas de control se encuentran programados en el DSP (Figura 2-10). Este es un dispositivo programable que cuenta con varios puertos de salida y entrada tales como: PWM, comunicación serial RS232, comunicación CAN (*Controller Area Network*), entradas analógicas de tensión, un puerto I2C y puertos de salidas digitales de propósito general (GPIO). Este dispositivo es un procesador muy versátil que se programa en lenguaje C, utilizando el software *Code Composer Studio* (CCS) de *Texas Instruments*. En particular se utilizó el dispositivo TMS320F8335.

Figura 2-11: Single Board Controller (SBC) [3].

El DSP se comunica por la interfaz de usuario en la cual se ingresan las consignas y se despliega la información relevante. Esta se encontraba montada sobre una SBC (Figura 2-11), la cual cuenta con un mini computador embebido con un sistema operativo Linux Debian. Cuenta con un procesador ARM, pantalla táctil, tres puertos seriales RS232 y RS485, dos entradas USB, un slot SD y un puerto I2C [3].

2.4. Comunicación Serial

La unidad de procesamiento de la central cuenta con varias formas de transmitir mensajes hacia los dispositivos que lo rodean. En particular, en esta aplicación se utilizóla comunicación serial para enlazar el DSP con la HMI. El DSP tiene dos módulos de comunicación serial, el módulo SPI (*Serial Peripheral Interface*), el cual es un módulo de comunicación síncrona que requiere que los dispositivos que se comunican por este medio estén sincronizados. Este tipo de comunicación permite grandes tasas de transferencia de información, sin embargo, tiene un alcance limitado, de no más de 20 cm de separación entre los dispositivos, por lo que está

pensada para comunicar dispositivos que se encuentran en una misma placa o en placas adyacentes [7].

El segundo módulo de comunicación serial es el SCI (*Serial Communications Interface*), de tipo asíncrona y relativamente lenta, sin embargo, es uno de los protocolos existentes de comunicación más simples, por lo que es ampliamente utilizado en la industria para comunicar distintos dispositivos. Además, puede alcanzar distancias mayores (mayor a 200 m) que la comunicación por SPI. Este tipo de puerto también se suele llamar UART (*Universal Asynchronous Receive and Transmit*), y cuenta con varios estándares, tales como el R232 y RS485. En particular, el estándar RS232 es el que se utiliza para la comunicación entre el DSP y la interfaz gráfica, debido a su sencillez y a que solo se requiere comunicación de punto a punto a distancias relativamente cortas.

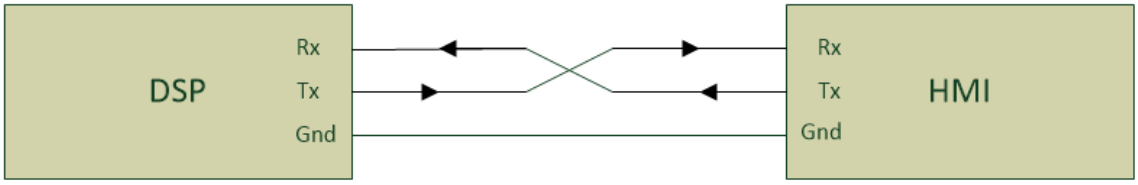


Figura 2-12 Conexiones de comunicación SCI RS232.

La comunicación por SCI consta de tres conexiones entre los dispositivos (un esquema se muestra en la Figura 2-12), dos de las cuales son para transmisión de datos y la tercera actúa como referencia común (Tierra común entre los dispositivos). Los canales de transmisión son Rx y Tx, que se ocupan de la recepción y transmisión de datos respectivamente. Para que la comunicación sea posible se requiere que el puerto Rx de uno de los dispositivos se encuentre conectado al puerto Tx del otro y viceversa, de modo que se encuentran cruzados. Por estos canales circula un tren de pulsos de amplitud +5V y -5V. Como medida de seguridad este protocolo nunca mantiene las líneas de comunicación en 0V, esta medida se conoce como Sin Retorno a Cero (NRZ, *No Return to Zero*). Esto permite detectar fallas de comunicación, en el caso de desconexiones o cortes.

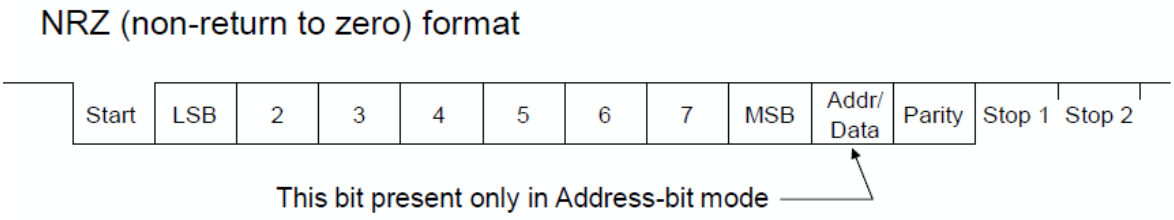


Figura 2-13: Estructura del mensaje en el protocolo SCI RS232 [7].

En la Figura 2-13 se muestra la estructura de un mensaje. Cada mensaje comienza con un bit de inicio de transmisión, lo siguen los bits de mensaje que pueden variar de 7 bit a 8 bits de

largo y puede agregarse un bit de paridad para añadir seguridad al mensaje, y finalmente uno o dos bits de fin de transmisión.

La comunicación serial por SCI RS232 está descrita por los siguientes parámetros: *Baud Rate*, Paridad y Número de bits de parada. El *Baud Rate* corresponde a la frecuencia de transmisión de datos y se mide en bits/s. La paridad corresponde a la inclusión o no inclusión del bit de paridad. Si se habilita esta restricción de seguridad, los bits de datos requieren cumplir la paridad indicada en el bit de paridad del mensaje, de lo contrario, el mensaje no es válido y se desecha. Finalmente el número de bits de parada indica si el mensaje tiene uno o dos bits para indicar el fin de transmisión.

El DSP cuenta con un mecanismo de FIFO (First In First Out) para el envío y recepción de datos. Este mecanismo permite ingresar los datos en un buffer de salida y una vez que este tiene el número configurado de datos guardados los envía todos de manera consecutiva. Similarmente, existe un buffer de entrada, el cual genera una interrupción cuando se ha alcanzado la cantidad requerida en el buffer de entrada. Por ejemplo, si se configura un FIFO de nivel 5 en la salida y uno de nivel 4 en la entrada, se esperaría a que se cargaran 5 datos en el buffer de salida para dar comienzo a la transmisión, y se esperaría recibir 4 datos antes de generar la interrupción de recepción.

2.5. Interfaz Hombre-Maquina

2.5.1. Interfaz

Una interfaz es un componente de un sistema que le permite al usuario interactuar con éste de manera física o cognitiva. En general, existen dos tipos de interfaces: las que despliegan información a través de luces, imágenes, sonidos, etc., y las que reciben información como: palancas, botones, teclados, pedales, etc. La primera categoría se le suele referenciar como indicadores, mientras que la segunda como controles o mandos. Estas interacciones forman un bucle cerrado en el cual el usuario ingresa información al sistema a través de los controles, y el sistema reacciona con la información. Finalmente, despliega una actualización de la información del sistema en los indicadores, para que el usuario pueda tomar nuevas decisiones y volver a ingresar nueva información al sistema mediante los controles [8].

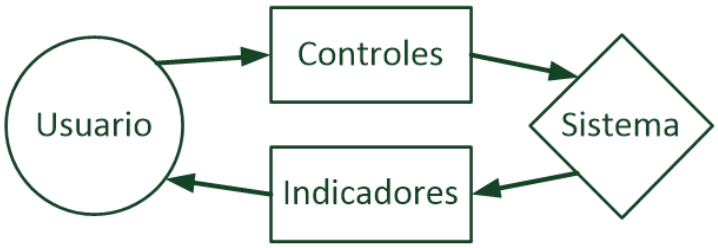


Figura 2-14: Ciclo de la relación entre usuario y sistema.

Como se observa en la Figura 2-14, la relación entre el usuario y la máquina forma un ciclo de retroalimentación. La información que entregan los indicadores puede ser clasificada en

distintos tipos: cualitativa (de estado de un sistema ej.: modo isla, detenido, sincronizado, etc.), cuantitativa (variable numérica: tensión, corriente, etc.), de chequeo (luz verde, roja) y de conocimiento de la situación (dar un sentido global al sistema ej.: imagen del sistema).

En particular, la interfaz utilizada en el proyecto es un híbrido de un indicador con un control, ya que una pantalla táctil permite entregar y recibir información. Además, ésta permite entregar una amplia gama de señales visuales y auditivas, así como permite presentar botones, cuadros de texto, diales, etc.

2.5.2. Usuario

Es fundamental analizar al usuario en el momento de desarrollar una interfaz hombre – máquina, dado que debe entregar la información de manera correcta al usuario, y recoger las señales de control. Por lo tanto, es primordial analizar la manera en que se relaciona con la HMI. Se pude decir que existen tres principios en los que se basa la relación entre el usuario y la HMI. Estos son: principio de procesamiento, representación y limitación de capacidad de procesamiento [8].

El primer principio se puede dividir en dos temas: recepción de la información y procesamiento de la información. La recepción de la información pasa por tres etapas: adquisición, retener y seleccionar, y finalmente, recuperación de la información deseada. La adquisición consiste en utilizar los distintos sentidos para captar la información que nos rodea. La etapa de selección y retención se encarga de escoger parte de la información captada y luego memorizarla mediante alguna estrategia como: repetición, asociación, etc. Finalmente, en la etapa de recuperación consiste en buscar una determinada información que ha sido guardada previamente.

Por otro lado, el procesamiento de la información consiste en la formación de una representación interna de la información, basándose en una serie de principios definidos. Estos principios corresponden a: detección de bordes, diferenciación de figura-fondo, y agrupación. La detección de bordes consiste en la conexión de elementos que presentan alguna uniformidad respecto a alguna característica (color, textura, movimiento, etc.). La diferenciación entre figura y fondo es el proceso mediante el cual, se determina que parte de la información es relevante (figura) y que parte no (fondo), usualmente el fondo ni siquiera es recordado posteriormente. Finalmente, la agrupación consiste, en asociar distintas partes de la información a una misma tarea o tema. Para esto el cerebro se basa en: proximidad, similitud, destino común, continuación, cierre, sincronía, región común y conexión entre elementos.

El principio de representación ilustra la organización de la información de la memoria, es mediante esta organización que se puede recuperar información. Una de las formas en la que se almacena esta información corresponde a los modelos mentales. Esto corresponde a la forma en como una persona almacena el comportamiento de un sistema. Estos modelos permiten realizar simulaciones internas del sistema, que el usuario utiliza para interactuar con el sistema. A medida que el usuario interactúa con el sistema el modelo mental se perfecciona. Sim embargo, estos modelos presentan ciertas limitaciones: son incompletos, pueden confundirse con modelos similares de otros sistemas y las simulaciones internas que se realizan no son necesariamente correctas.

Finalmente, el tercer principio postula que el sistema cognitivo de una persona es limitado y por tanto, no es capaz de procesar toda la información disponible. Además, la eficacia de estas tareas varía según la complejidad, los conocimientos y experiencia de la persona.

2.5.3. Ergonomía Cognitiva

La ergonomía cognitiva es: "el estudio de todas las actividades humanas relacionadas con el conocimiento y el procesamiento de la información que influyen o están influenciadas por el diseño de máquinas y objetos que usan las personas"[1]. El principal objetivo de esta disciplina en el desarrollo de interfaces, es mejorar las teniendo en cuenta al usuario como parte principal del diseño y como este entiende el funcionamiento del mismo, dando como resultado interfaces más seguras, eficaces y simples.

Estos resultados se deben a que el diseñador posee mayores conocimientos de cómo funciona el sistema, por lo que puede llegar a generar un modelo conceptual de la interfaz más complejo de lo requerido. Por el contrario, el usuario no necesariamente conoce el funcionamiento del sistema, y por tanto genera un modelo mental de la interfaz a medida que la utiliza y, por lo tanto, es bastante incompleto. Además, si la interfaz posee una complejidad superior a la necesaria esto repercutirá en la dificultad que el usuario tendrá en la creación de este modelo, y por tanto, será más propenso a cometer errores en la operación del sistema. Es por esto que el usuario debe ser tomado en cuenta al momento de diseñar una interfaz. En general se busca facilitar que el usuario forme un modelo mental de la interfaz, parecido al modelo del diseñador.

Para diseñar una interfaz eficaz se deben tener en cuenta los tres principios del sistema cognitivo. Podemos valernos de la forma en que se procesa la información recibida por los sentidos, por ejemplo, si se desea resaltar alguna información, se debe procurar un buen contraste con el fondo o valerse de la proximidad del indicador con el usuario, de modo que este indicador resalte en su campo visual. Estas acciones se valen de las etapas de detección de bordes y de diferenciación de fondo del procesamiento de los sentidos.

Otra forma es ocupar la agrupación, para que ciertos controles o indicadores se perciban como un conjunto. Esta técnica se utiliza para demarcar tareas, por ejemplo, si la acción sobre uno de los controles depende de los datos mostrados por dos indicadores, es lógico que estos dos últimos se encuentren próximos al control en cuestión. Existen distintas formas de agrupación: por proximidad, similitud, región común, sincronía, destino común, conexión entre elementos, etc.

Para ayudar a que el usuario genere un buen modelo mental, se debe proporcionar un buen modelo conceptual, en el que se puedan relacionar claramente las acciones en los controles con los cambios en el sistema, y se proporcione la información de manera coherente. También es recomendable desplegar un modelo simplificado del sistema que entregue información general sobre el estado de este. Finalmente, el tercer principio indica, que el usuario no es capaz de procesar una gran cantidad de información, por lo que no se debe recargar al usuario con

[1] Definición de ergonomía cognitiva [8]

información o controles irrelevantes para realizar una tarea, y además, estas últimas no deben requerir un procesamiento complicado de la información recibida.

2.5.4. Criterios a tener en cuenta al momento del diseño.

En este trabajo se desea diseñar una interfaz gráfica en un ordenador similar a una página web. A continuación se presentan consejos de diseño de interfaces. Estos criterios fueron desprendidos de los conceptos de ergonomía cognitiva [8], y de una guía de evaluación de páginas web [9]. A continuación, se detallan una serie de puntos a tener en cuenta al momento del diseño.

1. Mantener al usuario informado del estado del sistema, y siempre presentar un feedback cuando este realiza alguna acción.
2. Utilizar la lengua del usuario.
3. Mantener la sensación de control del usuario, presentar siempre vías de **retroceso** y **ayuda**. Además, mantener una consistencia en el formato de presentación de información y enlaces.
4. **Diseñar en función de modo de evitar que el usuario pueda cometer errores.** Minimizar la aparición de mensajes de errores.
5. Mantener opciones de navegación a la vista, e informar siempre al usuario que página de la interfaz se está observando actualmente.
6. **No mostrar información irrelevante o que no se necesite a simple vista.**
7. Simplificar estructuras de tareas que deba realizar el usuario. Por ejemplo, mantener agrupados informaciones y controles de una determinada tarea.
8. Mostrar claramente las acciones posibles en cada momento.
9. Esconder información y controles irrelevantes cuando sea necesario realizar un tipo de tarea.

2.5.5. QTCreator

QT es un programa de desarrollo de Aplicaciones basado en C++, con algunos elementos de JavaScript, que permite desarrollar aplicaciones sencillas rápidamente debido a que consta de una biblioteca con objetos predefinidos, tales como botones, etiquetas, cuadros de texto, etc. El programa de desarrollo se denomina QTCreator y está disponible de manera gratuita en la página QT-Project [10]. Además permite generar aplicaciones para múltiples sistemas operativos tales como Linux, Windows o Android, lo que le confiere una flexibilidad al desarrollador al convertir rápidamente una aplicación hecha para Linux a Windows o a Android. Otra ventaja significativa de QT, es el hecho de que esté basado en C++, ofreciendo a QT una amplia gama de foros en los que investigar en caso de tener dudas, como qtcenter.org o stackoverflow.com.

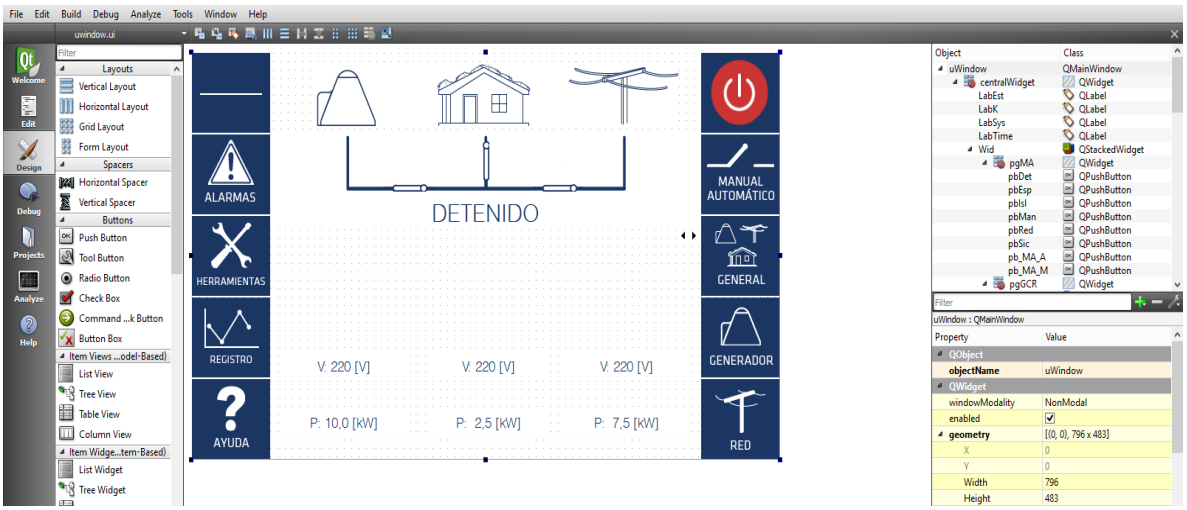


Figura 2-15: Entorno de diseño de QTCreator.

En la Figura 2-15 se muestra el entorno de desarrollo. Al lado izquierdo cuenta con una barra de herramientas, en la cual se pueden seleccionar distintas vistas del proyecto (edición de código, gráficos, menú de ayuda, etc.). Luego, cuenta con una barra de objetos gráficos, de la cual se pueden arrastrar botones, etiquetas, imágenes, etc. hacia la interfaz que ocupa la parte central de la pantalla, mientras que a la derecha se pueden ver las propiedades de los elementos que se encuentran en la interfaz.

Capítulo 3. Propuesta de HMI y Sistema de Manejo de Energía

3.1. Descripción General

En este trabajo se aborda el desarrollo de un prototipo de HMI y sistema de manejo de energía, para la Central Micro Hidráulica Plug & Play. El sistema de manejo de energía es el principal sistema de control de la unidad, actuando como supervisor de los distintos sub sistemas de control de la central y modificando las referencias de los distintos controladores cuando la unidad requiere cambiar de estados de operación.

En general, los estados dependen de la forma en que la unidad se conecta con los elementos del sistema, el empleo de distintas formas de control y la existencia de flujo de agua. Bajo esta perspectiva se identifican cuatro estados básicos: detenido, en espera, modo isla y sincronizado. Se debe destacar que los estados se encuentran relacionados por las transiciones, las cuales corresponden a secuencias de acciones que se deben tomar para transitar de un estado a otro.

Las transiciones se pueden separar en dos grupos: normales o de falla. Las transiciones normales se pueden producir debido a la acción de un usuario o al sistema de operación automática de la central. En particular, este último sistema no se encuentra desarrollado y se escapa de los alcances de este trabajo. Por otra parte, las transiciones de falla se disparan automáticamente cuando se rompe una de las condiciones normales de operación.

Las transiciones normales pueden ejecutarse por órdenes del usuario, requiriéndose la existencia de un canal de comunicación entre la HMI y el procesador central de la unidad. Este canal corresponde a una conexión serial RS232 entre el DSP y la HMI, por el cual se intercambian mensajes que transportan la información del estado de operación de la máquina, observaciones de las variables eléctricas del sistema, y órdenes de cambio de modo de operación.

La información recibida desde la HMI es desplegada en distintas pestañas, que agrupan la información según su fuente, por ejemplo: general, alarmas, operación, ayuda, etc. Estas pestañas conforman el modulo gráfico de la aplicación montada en la HMI. Específicamente, la aplicación fue desarrollada en QT, debido a que provee bloques de componentes gráficos que facilitan la creación de módulos como el descrito anteriormente.

Durante el resto del capítulo se presentan tres secciones. La primera trata los criterios y requerimientos que deben cumplir el sistema de manejo de energía y la aplicación de la HMI. Esto aborda: estados básicos, ex portabilidad, criterios de ergonomía cognitiva, etc. En la segunda sección se expone la implementación de ambos sistemas. Finalmente, en la tercera sección, se muestran las pruebas con las que se validaron ambos sistemas.

De modo general, se requiere que la central sea capaz de mantener cuatro estados de operación (Detenido, En espera, Sincronizado y Modo Isla), y que sea capaz de transitar de un estado a otro cuando se requiera. Además, la interfaz requiere que sea capaz de desplegar información referente al funcionamiento del sistema. Por último, es necesario que ambos

sistemas sean compatibles el uno con el otro, y puedan ser exportados a otras unidades similares.

3.2. Especificaciones y Criterios a Cumplir

Como se ha visto anteriormente, este trabajo aborda parte del desarrollo de dos sistemas esenciales en la Central Micro Hidráulica Plug & Play: el sistema de manejo de energía, y el desarrollo de la interfaz que media entre el usuario y la máquina. Ambos sistemas deben cumplir ciertos objetivos, los cuales se abordaran en esta sección. En particular, ambos sistemas deben ser flexibles, o en otras palabras, permitir una fácil adaptación si cambia la estructura del sistema.

El sistema de manejo de energía corresponde a un control que se encarga de supervisar el correcto funcionamiento de los distintos sub sistemas de la central. Debe mantener la unidad operando de forma estable, o guiar a la central hasta otro estado de operación, conforme las exigencias del usuario o las condiciones del sistema. Se pueden definir cuatro formas de operación, las cuales se revisan a continuación.

El primer estado consiste en un modo de "hibernación" del sistema, donde la unidad se encuentra desconectada de la carga local y la red eléctrica y la turbina se encuentra detenida. Sin embargo, los sistemas de control se encuentran energizados y listos para la operación. Este estado de operación se le denomina estado detenido.

En el segundo estado, la unidad se encuentra operando en vacío a una tensión y frecuencias determinadas por este control, cuyo objetivo final es llevar a la unidad a las mismas condiciones de operación de la red: $220V_{fn}$ y 50Hz. Estas condiciones se deben poder mantener de forma indefinida. En adelante, este estado se denomina estado en espera.

El tercer estado de operación (Isla), se encarga de controlar la unidad cuando esta ópera de forma aislada a la red. En otras palabras la unidad y la carga local, están conectadas entre sí y a su vez separadas de la red eléctrica. En este estado se debe mantener una operación a $220V_{fn}$ y 50Hz.

Finalmente, en el último estado, la central y la carga local se encuentran conectadas a la misma barra, por lo tanto, la tensión de generación y la frecuencia se encuentran determinadas por la red, de modo que se intercambia el control de tensión por el control de reactivos. Este modo se denomina estado sincronizado.

En conjunto con estos estados de operación se requieren transiciones para migrar de una forma de operación a otra. Estos métodos no solo se deben encargar de abrir y cerrar los contactores correspondientes, sino que deben verificar las condiciones necesarias para pasar de un modo de operación a otro, por lo que de manera básica, estas transiciones constan de tres pasos: primero verificar condiciones de operación, luego abrir o cerrar los contactores correspondientes, y finalmente activar o desactivar controles de tensión, factor de potencia o frecuencia. Otro aspecto relevante sobre las transiciones es el origen de éstas, ya que pueden provenir desde órdenes del usuario o fallas del sistema, por lo que se requiere una forma de identificar cada transición con base a un código único.

Este sistema de control - supervisor se comunica con el usuario a través de la interfaz gráfica, la cual debe representar al sistema de forma sencilla y entregar la información más relevante. En otras palabras, el usuario debe ser capaz de obtener una idea clara del estado de operación del sistema de forma rápida, de modo que la información más indispensable debe estar siempre a la vista y ser fácil de interpretar.

Otro aspecto clave en el diseño de una interfaz, es lograr que el usuario genere un modelo mental de comportamiento de la interfaz y el sistema. Bajo este marco, se requiere que la interfaz mantenga una estructura relativamente constante, siendo fácil de identificar qué información se entrega y como se debe proceder si se quiere observar otro tipo de información. Por tanto, el menú de pestañas debe permanecer visible en cada momento. Por otro lado, para facilitar la creación del modelo del sistema en sí, la HMI debe proveer una imagen que refleje la estructura y el estado de operación del sistema. Además, la HMI debe mostrar variables como la tensión de generación, consumo y red, la potencia activa total que fluye por cada parte del sistema, un detalle por fases para el generador y la red, y una zona en donde se muestren alarmas generadas por fallas en el sistema.

Por último, la interfaz debe proveer una zona donde el usuario pueda desencadenar una migración del estado de operación de la máquina, o cambiar la consigna de operación de manual a automático y viceversa. Además, se deben deshabilitar las transiciones irrealizables desde el estado actual de la máquina, para prevenir errores de operación o pérdida de la sensación de control por parte del usuario.

3.3. Implementación

Antes de implementar el control supervisor se realizó un análisis del comportamiento esperado de la central, que como se vio en la sección anterior debe ser capaz de transitar entre al menos cuatro estados básicos: Detenido, En Espera, Modo Isla y Sincronizado. Sin embargo, cabe la pregunta ¿Qué pasa si por un periodo de tiempo limitado, no es posible permitir la circulación de agua por el sistema? Esta situación puede ser provocada, por ejemplo, por limpieza del desarenador. Debido a esto, se agregan dos nuevo estados de operación posibles: Solo Red, y Mantención.

En dichos estados el consumo local se encuentra abastecido por la red eléctrica evitando una eventual pérdida de suministro. En particular, en el estado mantención, no existe flujo de agua por la turbina mientras que en el estado solo red la central se encuentra operando, e intentando lograr condiciones normales de operación a $220V_{fn}$ y 50Hz. Por esta razón, esto que finalmente se implementaron seis estados:

Detenido: la máquina se encuentra detenida ya que la válvula de mariposa se encuentra cerrada. Además, el generador está desconectado del consumo local y la red, y el consumo local también se encuentra desconectado de la red.

En espera: en este estado la máquina se encuentra operando, por lo que la válvula de mariposa se encuentra completamente abierta. Sin embargo, el generador, la red y el consumo se encuentran desconectados entre sí.

Sincronizado: la central se encuentra conectada con la red y el consumo, operando la tensión y frecuencia de la red

Isla: la central se encuentra conectada al consumo local y desconectado de la red. Las consignas de tensión y frecuencia son fijadas por el generador.

Red: el estado Red corresponde a la situación en que el generador se encuentra operando, sin embargo, se encuentra desconectado de la red y el consumo, por lo que este último está siendo alimentado por la red.

Mantención: el generador se encuentra detenido, la válvula está cerrada y está desconectado de la red y el consumo. Este último está siendo alimentado por la red.

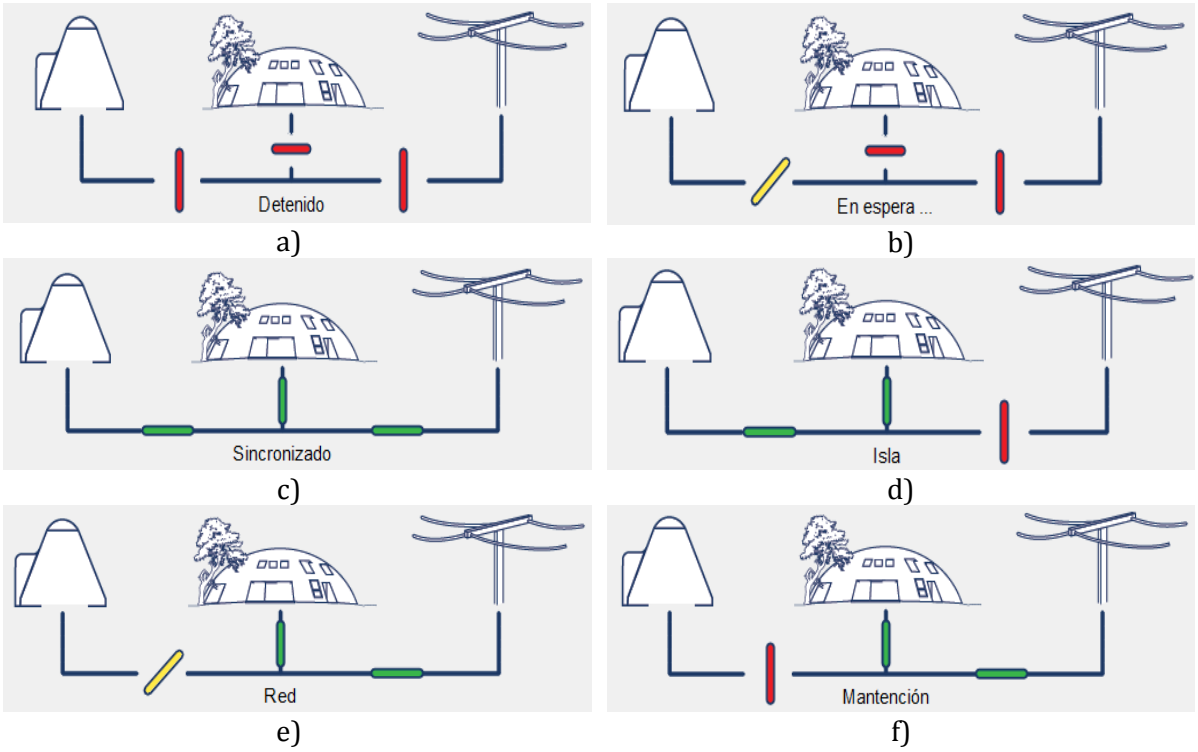


Figura 3-1: Estados de operación.

Además de los estados mencionados, cabe la posibilidad de agregar un estado sincronizado puro, en el cual el generador y la red se encuentran conectados y aislados del consumo local. Sin embargo se considera que esta situación corresponde a un caso particular del estado Sincronizado ya que no se requiere la realización de ningún cambio por parte de la unidad de generación.

En general las únicas diferencias entre los estados corresponden a los tipos de controles que se ejecutan y sus referencias, siendo la salvedad los estados En espera y Solo Red. En estos, además de configurar los controles de tensión, factor de potencia y frecuencia, se agrega un segundo lazo al control de tensión. Este consta de un control V/F que busca que la tensión aumente proporcionalmente con la frecuencia del sistema. Este aumento se debe a que en estos estados el flujo de agua no es necesariamente constante debido a que la central se encuentra

comenzado a operar o en rumbo a detenerse. A continuación se detallan las consignas utilizadas en cada estado.

1) Detenido:
   a) Mantener la válvula de mariposa cerrada.
   b) Control de campo desactivado (corriente de campo en cero).
   c) Referencia de tensión cero.
   d) Contactor generador abierto.
   e) Contactor red abierto.
   f) Contactor carga abierto.
   g) No sincronizado. (señal al control de frecuencia)
   h) Control de frecuencia desactivado. (señal al control de frecuencia)

2) En Espera:
   a) Mantener la válvula de mariposa abierta.
   b) Control de campo activado en modo control de tensión.
   c) Referencia de tensión dependiente de la frecuencia del generador.
   d) Contactor generador abierto.
   e) Contactor red abierto.
   f) Contactor carga abierto.
   g) No sincronizado.
   h) Control de frecuencia activado. (a partir de 30 V)

3) Sincronizado:
   a) Mantener la válvula de mariposa abierta.
   b) Control de campo modo factor de potencia.
   c) Referencia de $\varphi = 0.3175$ y $\cos(\varphi) = 0.95$.
   d) Contactor generador cerrado.
   e) Contactor red cerrado.
   f) Contactor carga cerrado.
   g) Sincronizado.
   h) Control de frecuencia activado.

4) Isla:
   a) Mantener la válvula de mariposa abierta.
   b) Control de campo modo control de tensión.
   c) Referencia de 220 V
   d) Contactor generador cerrado.
   e) Contactor red abierto.
   f) Contactor carga cerrado.
   g) No sincronizado.
   h) Control de frecuencia activado.

5) Red:
   a) Mantener la válvula de mariposa abierta.
   b) Control de campo activado en modo control de tensión.
   c) Referencia de tensión dependiente de la frecuencia del generador.
   d) Contactor generador abierto.
   e) Contactor red cerrado.
   f) Contactor carga cerrado.

g) No sincronizado.
h) Control de frecuencia activado. (a partir de 30 V)

6) Detenido:
   a) Mantener la válvula de mariposa cerrada.
   b) Control de campo desactivado.
   c) Referencia de tensión cero.
   d) Contactor generador abierto.
   e) Contactor red cerrado.
   f) Contactor carga cerrado.
   g) No sincronizado.
   h) Control de frecuencia desactivado.

Por otra parte, las transiciones corresponden a la forma en que la central migra de un estado a otro. Se suelen realizar cambios de referencia para los distintos controladores de la unidad, como también modificaciones en los estados del sistema. Sin embargo, antes de realizar algún cambio se deben comprobar que las condiciones necesarias para efectuar la migración son cumplidas.

Algunas de estas transiciones son bastante sencillas y rápidas de realizar, mientras que otras requieren condiciones muy específicas, y por tanto, realizarlas requiere de una mayor cantidad de tiempo. Por ejemplo, para transitar de modo En Espera a modo Isla se requiere, primero, verificar las condiciones de operación del generador. Si estas son adecuadas, entonces se procede a cerrar el contactor ubicado entre el generador y el consumo local. Finalmente, se espera la señal de verificación de cierre de contactor, y el sistema se encuentra en modo isla. Todo esto tarda menos de un segundo en ejecutarse. Por otro lado, una transición al estado de Sincronizado, requiere verificar que la unidad y la red operen a una tensión similar, con menos de 10V de diferencia, y que además operen a la misma frecuencia y fase durante un periodo de tiempo determinado. Dado que la unidad no cuenta con un control de fase, se requiere esperar que las tensiones de ambos sistemas se pongan en fase de manera natural, proceso que puede demorar varios segundos.

Además del tiempo de ejecución, las transiciones se pueden clasificar según su origen. Existen tres sucesos que pueden disparar una transición: la orden de un usuario, la inteligencia de la central o una situación de falla. Las dos primeras fuentes generan transiciones normales mientras que la tercera genera transiciones de falla, que a su vez se pueden clasificar en fallas leves y graves. Algunas de estas transiciones de falla, provocan que el generador transite a uno de los dos posibles estados seguros: Detenido y Mantenimiento.

Para diferenciar las transiciones, a cada una se le asigna un código único, el cual está compuesto por dos caracteres y dos dígitos. En el caso de las transiciones nórmales las dos primeras letras son “TN”, mientras que en el caso en que la transición sea de falla, el primer carácter depende del carácter de la falla. Si es leve, entonces corresponde la letra “L”, y por el contrario, si es una falla grave, corresponde la letra “G”. La segunda letra en caso de falla, depende de la causa de esta. Por ejemplo, si la falla fue motorización, la segunda letra será “M”. En el caso de los dígitos, cualquiera sea la causa de la transición, el primero corresponde al estado inicial de la máquina al efectuarse la transición, mientras que el segundo corresponde al estado final de la máquina. Este código se puede utilizar para la creación de un registro de eventos que se puede guardar en la HMI, y además forma la base para la comunicación de los mensajes de falla con esta.

Tabla 3.1 Transiciones posibles

Transiciones				
ID	**Tipo**	Desde	**Hasta**	**Descripción**
TN12	**Normal**	**Detenido**	**En espera**	**-**
TN15	**Normal**	**Detenido**	**Red**	**-**
TN16	**Normal**	**Detenido**	**Mantención**	**-**
TN21	**Normal**	**En espera**	**Detenido**	**-**
TN23	**Normal**	**En espera**	**Sincronizado**	**-**
TN24	**Normal**	**En espera**	**Isla**	**-**
TN25	**Normal**	**En espera**	**Red**	**-**
TN26	**Normal**	**En espera**	**Mantención**	**-**
TN41	**Normal**	**Isla**	**Detenido**	**-**
TN42	**Normal**	**Isla**	**En espera**	**-**
TN43	**Normal**	**Isla**	**Sincronizado**	**-**
TN31	**Normal**	**Sincronizado**	**Detenido**	**-**
TN32	**Normal**	**Sincronizado**	**En espera**	**-**
TN34	**Normal**	**Sincronizado**	**Isla**	**-**
TN35	**Normal**	**Sincronizado**	**Red**	**-**
TN36	**Normal**	**Sincronizado**	**Mantención**	**-**
TN51	**Normal**	**Red**	**Detenido**	**-**
TN56	**Normal**	**Red**	**Mantención**	**-**
TN53	**Normal**	**Red**	**Sincronizado**	**-**
TN61	**Normal**	**Mantención**	**Detenido**	**-**
TN65	**Normal**	**Mantención**	**Red**	**-**
GM31	**Falla**	**Sincronizado**	**Detenido**	**Motorización**
GM36	**Falla**	**Sincronizado**	**Mantención**	**Motorización**
GQ34	**Falla**	**Sincronizado**	**Isla**	**Calidad de suministro**
GQ35	**Falla**	**Sincronizado**	**Red**	**Calidad de suministro**
GQ33	**Falla**	**Sincronizado**	**Sincronizado (sin carga)**	**Calidad de suministro**
GC36	**Falla**	**Sincronizado**	**Mantención**	**Comunicación**
GC61	**Falla**	**Mantención**	**Detenido**	**Comunicación**
GC41	**Falla**	**Isla**	**Detenido**	**Comunicación**
GC63	**Falla**	**Mantención**	**Sincronizado**	**Comunicación**

LA52	**Falla**	**Red**	**En espera**	**Ausencia Red**
LA34	**Falla**	**Sincronizado**	**Isla**	**Ausencia Red**
LA61	**Falla**	**Mantención**	**Detenido**	**Ausencia Red**
LS33	**Falla**	**Sincronizado**	**Sincronizado (bloquear isla)**	**Sobrecarga**
LS41	**Falla**	**Isla**	**Detenido**	**Sobrecarga**
LS42	**Falla**	**Isla**	**En espera**	**Sobrecarga**
LF33	**Falla**	**Sincronizado**	**Sincronizado (bloquear isla)**	**Falta recurso**
LF42	**Falla**	**Isla**	**En espera**	**Falta recurso**
LC35	**Falla**	**Sincronizado**	**Red**	**Falta campo**
LC42	**Falla**	**Isla**	**En espera**	**Falta campo**
LC56	**Falla**	**Red**	**Mantención**	**Falta campo**
LC21	**Falla**	**En espera**	**Detenido**	**Falta campo**

Cabe destacar que el comportamiento de las fallas es lo que le confiere parte de la "inteligencia" a la central, ya que van a ser estas la que eventualmente provoquen que la central se aislé cuando la red sale de operación.

Tanto las transiciones como los estados de operación forman parte de una interrupción que se ejecuta con una frecuencia de 4.7kHz, en la cual, primero, se toman medidas de las variables del sistema y luego se ejecutan acciones con base a estas. Sin embargo, antes de poder ejecutar estas funciones, se deben configurar los periféricos y sistemas que serán utilizados en el transcurso de la operación de la máquina.

El primer paso consta de configurar el reloj interno del DSP, con base al cual se configuran todos los demás relojes del sistema. En otras palabras, la frecuencia del PWM utilizado en el campo del generador, el reloj 1 que controla el envió de mensajes a la HMI, y la frecuencia con que se toman datos del sistema, dependen de este reloj. En particular, este opera a una frecuencia de 750 MHz. El siguiente paso es guardar el programa en la memoria flash del dispositivo, lo que permite que el programa se ejecute cada vez que el DSP se energice. Una vez que el programa ha sido guardado, se procede a asignarle una función a cada GPIO del DSP. En este caso, se asignancanales PWM y salidas digitales, mientras que el resto se dejan como entradas digitales.

Luego de configurar los GPIOs, se procede a configurar un reloj encargado de enviar información a la HMI. Este fue configurado con una frecuencia de 3,33Hz, ya que solo se desean enviar tres tipos de mensajes por segundo hacia la HMI. Después, se procedió a configurar los canales ADC del dispositivo, de manera que las mediciones se tomen de forma secuencial. Por otra parte se sincronizo el inicio de adquisición de datos con el PWM 5, el cual se configuró posteriormente a una frecuencia de 4.7kHz. Además de configurar el PWM 5, se habilito el PWM 1 para su utilización en el control de campo de la central, y se le asignó una frecuencia de operación de 3.75kHz. Luego, se procedió a configurar la comunicación serial, utilizando una tasa de 9600 bit/s, ocho bits de datos, un bit de fin de transmisión y sin paridad.

Finalmente, se procedió a configurar el sistema de interrupciones del dispositivo asignándole una interrupción a los siguientes eventos: inicio de toma de datos (PWM 5), término de toma de datos (interrupción por ADC), ciclo del reloj 1 (interrupción por tiempo), recepción de mensajes por el canal serial (SCI Rx), e inicio de envío de mensajes por el canal serial (SCI Tx). Luego de completar la configuración del dispositivo se procede a entrar en un ciclo infinito en el cual se ejecuta solo una de las llaves anti bloqueo de dispositivo. Existen dos llaves antibloqueo, si estas no son ejecutadas periódicamente el DSP procede a reiniciarse, la segunda llave antibloqueo es ejecutada en la interrupción desencadenada por el reloj 1.

El sistema de control supervisor de la central, se encuentra anclado a la interrupción de fin de toma de datos, esto se debe a que para ejecutar todas las acciones de control se requieren de nuevas muestras del sistema. Como se mencionó esta interrupción se dispara a 4.7kHz y posee dos formas de ejecución, la primera solo se realiza cuando el sistema se encuentra en modo detenido y se acaba de encender, de lo contrario se efectúa la segunda forma. A continuación se detallan estas dos formas:

- Interrupción por ADC offset: esta interrupción se dispara una vez que se han terminado de recoger los datos de los puertos ADC. Mediante esta interrupción se calcula los ceros de las variables eléctricas del sistema, para poder determinar el offset (desviaciones del cero) de las variables medidas por los transductores, es por esto que esta rutina solo se ejecuta 30.000 veces después de encender el sistema, los ceros del sistema se determinan mediante un filtro pasa bajo.

- Interrupción por ADC: al igual que la interrupción anterior ADC Offset, esta se ejecuta una vez que la mediciones se han recogido desde los puertos ADC, esta función solo se realiza una vez que se haya determinado el offset de las variables medidas. En esta función se ejecutan varios bloques de control y procesamiento de datos, los cuales se detallan más adelante.

La interrupción ADC es la que se encarga de procesar las variables y ejecutar los bloques de control del sistema, por lo que es la interrupción principal del sistema, mientras que las demás se encargan de proveer “servicios” a ésta. Durante esta interrupción se ejecutan las siguientes funciones:

- Rutina ADC: esta rutina se encarga de tomar los datos recogidos por los ADC (tensiones fase-fase y corrientes de línea) y procesarlos de manera de obtener tensiones de fase, tensiones rms, corrientes rms, potencia activa y reactiva, frecuencia eléctrica, ángulo de fase y factor de potencia. Esto se lleva a cabo mediante filtros, PLL, y transformadas de Clark y Park.

- Rutina de Fallas: una vez que se tienen los datos procesados se determina si existe alguna falla en el sistema, de existir se toman las medidas necesarias para corregirla o pasar a algún modo de operación seguro. A la Fecha se han implementado dos de las fallas: motorización y sobre tensión.

- Rutina Identificar: en esta rutina se determina si se ha ingresado una orden de cambio de estado ya sea por la HMI a través de la interrupción SCI Rx o por alguna falla en el sistema. En el caso de que se requiera transitar a otro estado se activa una de las

transiciones posibles y el sistema salta inmediatamente a la rutina de transiciones, de lo contrario se prosigue a la rutina de estados, y luego a la rutina de control.

- Rutina de Estados: la rutina de estados consiste en una secuencia de condiciones que verifican el estado actual y fijan las consignas para dicho estado. Por ejemplo, si el sistema se encuentra detenido se confirmarán las órdenes para mantener la válvula cerrada y la corriente de campo en 0A, también se reiterarán las órdenes para mantener los contactores abiertos. Más adelante se detallan las órdenes específicas para cada estado.

- Rutina de Transiciones: la rutina de transiciones solo permanece activada durante una o dos interrupciones de ADC, ya que en su mayoría solo se encargan de abrir o cerrar los contactores, siendo la excepción la transición de sincronización, la cual puede demorar alguno minutos.

- Rutina de Control: en esta rutina se ejecutan los cálculos relacionados con los controladores de tensión y factor de potencia.

- Rutina de Actuación: se envían la señales a los distintos actuadores del sistema: contactores y al circuito del chopper de campo.

El orden presentado fue creado con base al requerimiento de flexibilidad del sistema, en particular la interrupción por ADC dividida en módulos que poseen una función específica, lo que permite una fácil comprensión del código, e intercambiar módulos sin tener que realizar grandes cambios en el programa.

Las otras tres interrupciones corresponden a interrupciones de comunicación, que se encargan de recibir y procesar los mensajes desde la HMI, enviar mensajes o determinar qué mensaje corresponde enviar y dar inicio a la interrupción que los envía.

- Interrupción por tiempo: es básicamente una función que se ejecuta cada 300ms que se encarga de determinar el mensaje que debe ser enviado a la HMI del sistema, luego disparar la interrupción SCIA Tx y finalmente activar la segunda llave anti bloqueo del sistema.

- Interrupción por SCI Tx: esta interrupción simplemente transfiere el mensaje al bus de salida del puerto serial.

- Interrupción por SCI Rx: finalmente esta última interrupción se activa cada vez que se completa el buffer de entrada del puerto serial, y se encarga de descifrar el mensaje recibido y modificar las variables pertinentes del sistema.

El sistema de recepción de mensajes consta de un buffer de cinco caracteres que al llenarse activa la interrupción SCI Rx, para descifrar la información contenida y su posterior utilización en la interrupción por ADC.

Por otro lado se trasladó el sistema de la HMI a un computador debido a problemas técnicos presentados por la SBC. En particular ésta presento problemas de retardos significativos ante

una acción del usuario. Aun cuando se cambió hardware de la HMI, se optó por mantener QT como la plataforma de desarrollo, debido a la flexibilidad de esta plataforma.

La nueva aplicación fue construida con base a módulos encargados de realizar tareas específicas, buscando facilitar el diseño de posteriores versiones. La aplicación fue construida utilizando los siguientes módulos: un módulo de comunicación serial, y un módulo de retención de datos (variables del sistema) y de despliego gráfico. Además se pueden agregar los módulos de Historial y Alertas, los cuales guardan en un archivo de texto plano las distintas acciones o sucesos ocurridos en el sistema.

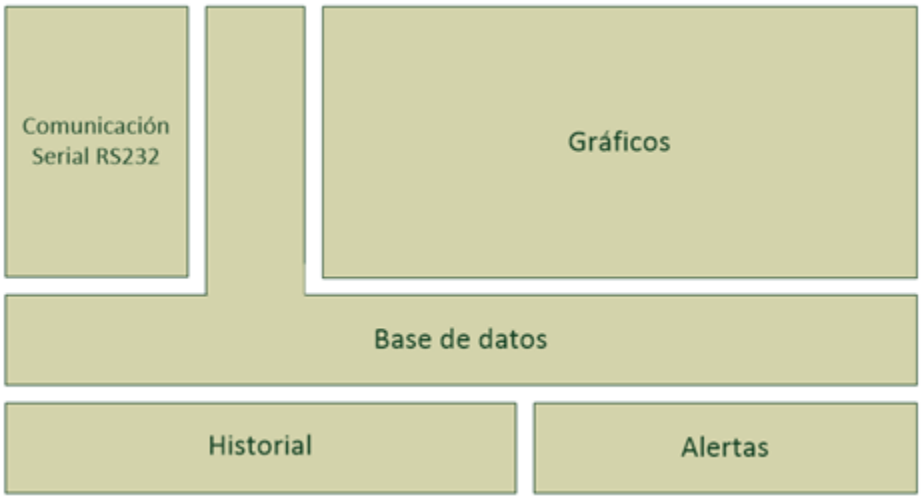


Figura 3-2: Diagrama de estructura de la aplicación.

En el diagrama de la Figura 3-2, cada bloque representa un módulo de la aplicación. El módulo RS232 se encarga de la comunicación con el DSP. El módulo de Base de datos es el que administra el guardado de datos y la comunicación entre los distintos módulos. El módulo de Gráficos se encarga de desplegar en pantalla el estado del sistema, también recibe las órdenes del usuario. Los módulos de Historial y Alerta se encargan de registrar en un archivo de texto los eventos que ocurren en el sistema, tales como cambios de consignas, intrusiones ingresadas por el operador, fallas ocurridas, transiciones realizadas.

En este orden de ideas, el módulo de comunicación serial presenta la mayor variabilidad, ya que a diferencia de los demás módulos que componen la aplicación, éste requiere la utilización del puerto serial del dispositivo por lo que si se desea montar la interfaz gráfica en otro dispositivo, se debe reescribir parte del módulo. Básicamente, se debe variar la forma en que se accede al puerto serial, ya que la configuración es estándar tanto para el DSP como la HMI. En particular se utiliza la siguiente configuración serial:

- Baud Rate: 9600.
- Bits de datos: 8.
- Bits de termino de mensaje: 1.
- Paridad: ninguna.

Para establecer un canal de comunicación serial entre el DSP y la HMI, se debe crear un objeto serial, con el cual la aplicación se comunica con el puerto del dispositivo. Este objeto

permite acceder a los registros de configuración del puerto, además realiza el envío y recepción de datos hacia y desde el DSP.

Una situación que hay que tener en cuenta al momento de diseñar una aplicación como ésta es la pedida del canal de comunicación. En este caso, se desea reiniciar el puerto serial, debido a esto se incorporó un sistema controlado por un reloj, que se encarga de contar el número de mensajes recibidos en un lapso de tiempo determinado, si este número es menor a un umbral, se procede a crear y configurar nuevamente el objeto que maneja el puerto serial.

Para controlar la recepción de mensajes desde el DSP se utilizó un sistema de buffer de entrada de 14 caracteres. Al completarse este buffer el sistema desencadena una interrupción en donde se procesa la información recibida por el DSP y luego se actualizan las variables en la base de datos de la HMI. Las interrupciones en esta plataforma cortan el hilo principal de ejecución, el cual se encarga de manejar los aspectos gráficos de la aplicación. Para evitar esta situación se decidió agregar un thread, que corresponde un hilo de ejecución paralelo al hilo principal, que posee un tiempo de vida limitado. Este thread es disparado por el mismo reloj que verifica el correcto funcionamiento del canal de comunicación. Tanto la función de envío de datos como la de recepción fueron ancladas a este thread. Este reloj se dispara cada 300ms aproximadamente.

Se definieron, ocho mensajes de 14 bytes de largo, cada mensaje comienza con carácter identificador y termina con los caracteres "\n" y "\r". Los mensajes se muestran en la Tabla 3.2

Tabla 3.2: Mensajes comunicación serial versión 1.

Mensajes Comunicación Serial								
Identificador	"T"	"P"	"F"	"E"	"G"	"R"	"C"	"A"
Dato 1	Vgen rms	Pgen	Fgen	Estado	Vgen AB	Vred AB	Vcon AB	"ID1"
Dato 2								"ID2"
Dato 3	Vcon rms	Pcon	Fcon	Estado Siguiente	Vgen BC	Vred BC	Vcon BC	"ID3"
Dato 4								"ID4"
Dato 5	Vred rms	Pred	Fred	Ref V	Vgen CA	Vred CA	Vcon CA	"x"
Dato 6								"x"
Dato 7	Igen rms	Qgen	"x"	Ref *cos(φ)*	Igen A	Ired A	Icon A	"x"
Dato 8			"x"					"x"
Dato 9	Icon rms	Qcon	"x"	Trans	Igen B	Ired B	Icon B	"x"
Dato 10			"x"					"x"
Dato 11	Ired rms	Qred	"x"	Válvula	Igen C	Ired C	Icon C	"x"
Dato 12			"x"	Kgcr				"x"
Termino 1	"\n"							
Termino 2	"\r"							

En la Tabla 3.2 se muestran la estructura de mensajes que se pueden enviar desde el DSP hacia la HMI, algunas de las variables que lo componen requieren algún tipo de pre procesamiento, por ejemplo amplificar una variable con componente decimal. Otro ejemplo es la forma en que se envía el estado de los contactores. Con el fin de reducir el estado de estos a una sola variable se implementó la siguiente función:

$$Kgcr = Kgen + 2 \cdot Kred + 4 \cdot Kcon \qquad 3.1$$

También cabe destacar que el mensaje de fallas corresponde a la última columna en la Tabla 3.2. En este mensaje la variable "ID1" corresponde a la gravedad de la falla, el segundo carácter "ID2" corresponde al tipo de falla, y finalmente las últimas dos variables "ID3" e "ID4" corresponden al estado en que ocurrió la falla y al estado que la central migrara para despejarla.

Por otro lado los mensajes desde la HMI se muestran en la Tabla 3.3.

Tabla 3.3: Estructura de mensajes desde la HMI al DSP.

Mensajes Comunicación Serial desde la HMI					
Identificador	"E"	"V"	"P"	"T"	"A"
Dato 1	Estado Siguiente	Referencia de tensión	Referencia de factor de potencia	Estado de comunicación	Posición de la válvula
Dato 2					
Termino 1	"x"				
Termino 2	"\n"				

Como se mencionó anteriormente los datos obtenidos desde el procesamiento de los mensajes recibidos desde ele DSP son almacenados en el módulo de base de datos. Este módulo consta de dos estructuras, la primera es un objeto donde se guardan las variables eléctricas del sistema: tensiones, corrientes, frecuencias, potencia, estado de operación, etc. En la segunda estructura se guardan las variables mecánicas del sistema como: la posición de la válvula y el nivel del estanque. Este módulo cuenta con dos mutex, uno para cada estructura de datos, teniendo la labor de evitar choques entre el thread de comunicación y el hilo principal de ejecución. Esto se realiza obligando a que cada vez que una función desea acceder a alguna de estas estructuras debe poseer la llave de acceso (el mutex). De esta manera se evitan choques de datos que pueden llevar a un colapso de la aplicación.

Otro de los módulos que componen esta aplicación corresponde a el modulo gráfico, encargado de la interacción con el usuario, para un correcto diseño se debe tener en cuenta las recomendaciones ofrecidas por la teoría de ergonomía cognitiva [8] y la guía de diseño de páginas web [9].

Recomendaciones:

1. Utilizar la lengua del Usuario.
2. Mantener al usuario informado del estado del sistema.
3. Mantener opciones de navegación a la vista, e informar siempre al usuario que página de la interfaz se está observando actualmente.
4. Mantener la sensación de control del usuario, presentar siempre vías de **retroceso** y **ayuda**. Además mantener una consistencia en el formato de presentación de información y enlaces.
5. **No mostrar información irrelevante o que no se necesite a simple vista.**

6. Simplificar estructuras de tareas que deba realizar el usuario. Por ejemplo mantener agrupados informaciones y controles de una determinada tarea.
7. Mostrar claramente las acciones posibles en cada momento.
8. Esconder información y controles irrelevantes cuando sea necesario realizar un tipo de tarea.
9. **Diseñar en función de modo de evitar que el usuario pueda cometer errores.** Minimizar la aparición de mensajes de errores.
10. Siempre presentar un feedback cuando este realiza alguna acción.

El primer punto es relativamente sencillo debido a que la unidad está pensada para ser comercializada en Chile, donde predomina el lenguaje español utilizado en la anterior HMI. El siguiente punto corresponde proveer al usuario de un "resumen" del sistema. Este objetivo se puede lograr mostrando en pantalla un esquema del sistema donde se mostrará como la unidad se encuentra conectada con los demás componentes: red y consumo local. Además, esta imagen del sistema puede ayudar a la construcción del modelo del sistema en la mente del usuario y del estado de contactores es importante estregar la forma de operación actual de la máquina.

Figura 3-3: Imagen del estado del sistema

En la Figura 3-3 se muestra el "resumen" del estado del sistema, se observa tres conectores conectados a un punto común, la posición de estos determina el estado de operación de la unidad.

El siguiente punto aborda el menú de navegación, se recomienda mantener este siempre a la vista para proveer al usuario un fácil acceso hacia las distintas vistas de la aplicación.

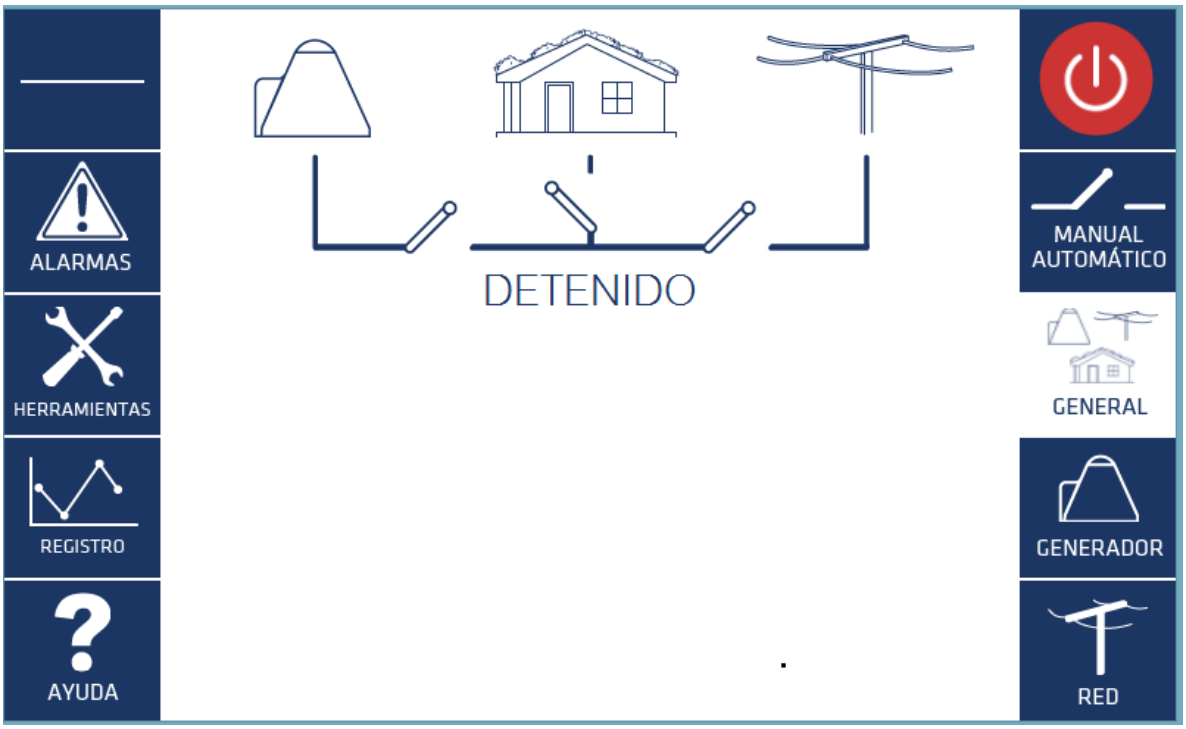


Figura 3-4: Estructura general de la interfaz.

En la Figura 3-4 se muestra el menú de navegación en los laterales izquierdo y derecho, en este menú cuenta con las siguientes opciones de navegación: alarmas, herramientas, registro ayuda, manual automático, general, generador y red. Finalmente en la esquina superior izquierda se muestra una barra horizontal en donde se ubicara la hora y fecha, mientras que en la esquina superior derecha se encuentra el botón de encendido y apagado de la central. En la imagen se observa una de las opciones resaltada con un fondo blanco, esta correspondería a la forma de indicar la pestaña actual que está observando el usuario.

Dado que el menú y el resumen del sistema deben permanecer a la vista del usuario en todo momento, se determinó que la región central inferior será la zona en la que se muestren las distintas pestañas de la aplicación: las cuales corresponden a las opciones de navegación antes expuestas. Esta estructura fue pensada para proveerle al usuario un formato consistente. Además, cada botón tiene función en forma de texto y de imagen descriptiva del tema de la pestaña a la que representa, ayudando al usuario a mantener una sensación de control del sistema.

Para determinar que pestañas se deberían agregar a la interfaz se pensó en la información debería estar disponible y que acciones puede tomar el usuario. La primara corresponde a un resumen un poco más detallado del sistema, se pensó incluir la tensión, frecuencia y potencia de cada uno de los componentes. Esta información estaría disponible en la pestaña general. Además de un resumen más acabado del sistema se incluyó un detalle por fases para el generador y la red, localizados en las pestañas de generador y red.

Otra información relevante es la activación de protecciones en la central, para esto se destinó la pestaña de alarmas en donde se muestran las tres últimas fallas, o alarmas ocurridas en el sistema. También se pensó en una pestaña más técnica en donde se mostraran, algunas

variables más específicas del sistema como el nivel del estanque, la posición de la válvula, el estado del canal de comunicación con el DSP, el nivel de carga de la batería del sistema, etc. Esta pestaña se les denominó herramientas.

Además de mostrar información, la interfaz debe ser capaz de recibir información por parte del usuario, por lo que se creó la pestaña de Manual y Automático. En esta pestaña el usuario puede ingresar consignas de operación: como el estado en que desea que la unidad opere, o la forma en que desea que se hagan las transiciones: manuales o automáticas, también se pensó en agregar referencias de tensión y factor de potencia.

Dado que tanto el usuario como la unidad pueden tomar decisiones de como operar, se agregó una pestaña de registro en donde se muestre las últimas acciones que han ocurrido en el sistema y la fuente de estas acciones: esto puede abarcar cambios de estados, operaciones realizadas por el usuario, alarmas etc.

Finalmente se agregó una pestaña de ayuda en la que se puede incluir una descripción del sistema, de los distintos estados de operación y sus diferencias, contactos, etc.

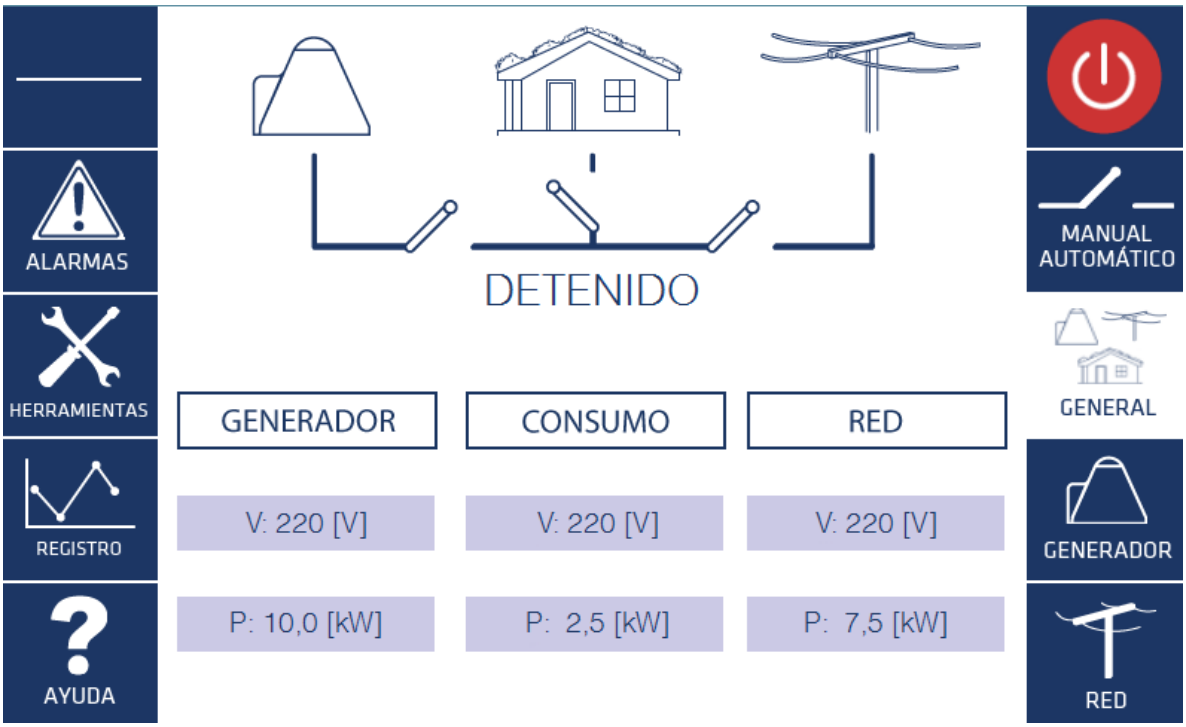


Figura 3-5: Estructura final, pestaña de general

En la Figura 3-5 se muestra la interfaz ya terminada. En esta versión no se muestra la frecuencia de los componentes ya que se consideró un dato muy específico. También se puede observar la disposición de la información, en esta pestaña se agruparon por columna cada uno de los componentes del sistema, siguiendo la pauta establecida por la imagen de resumen ubicada en la parte superior de la interfaz, esta agrupación espacial debería ayudar al usuario a relacionar la información con su respectivo componente.

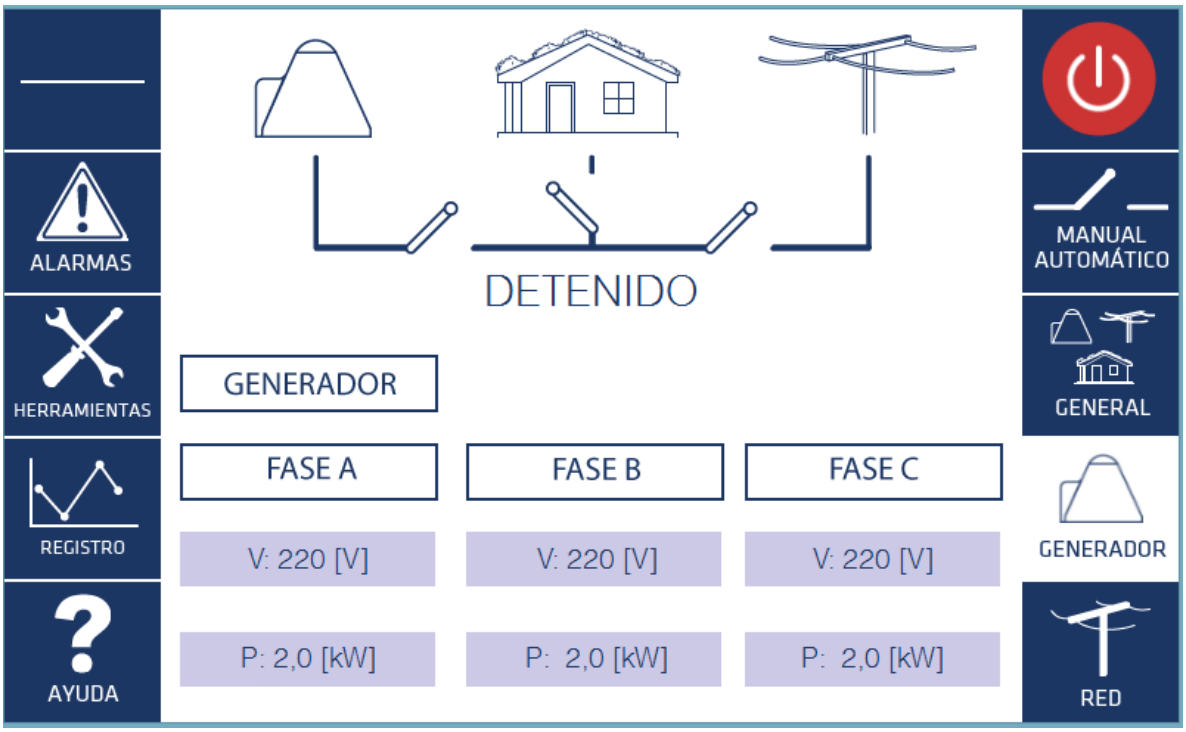


Figura 3-6: Pestaña del generador.

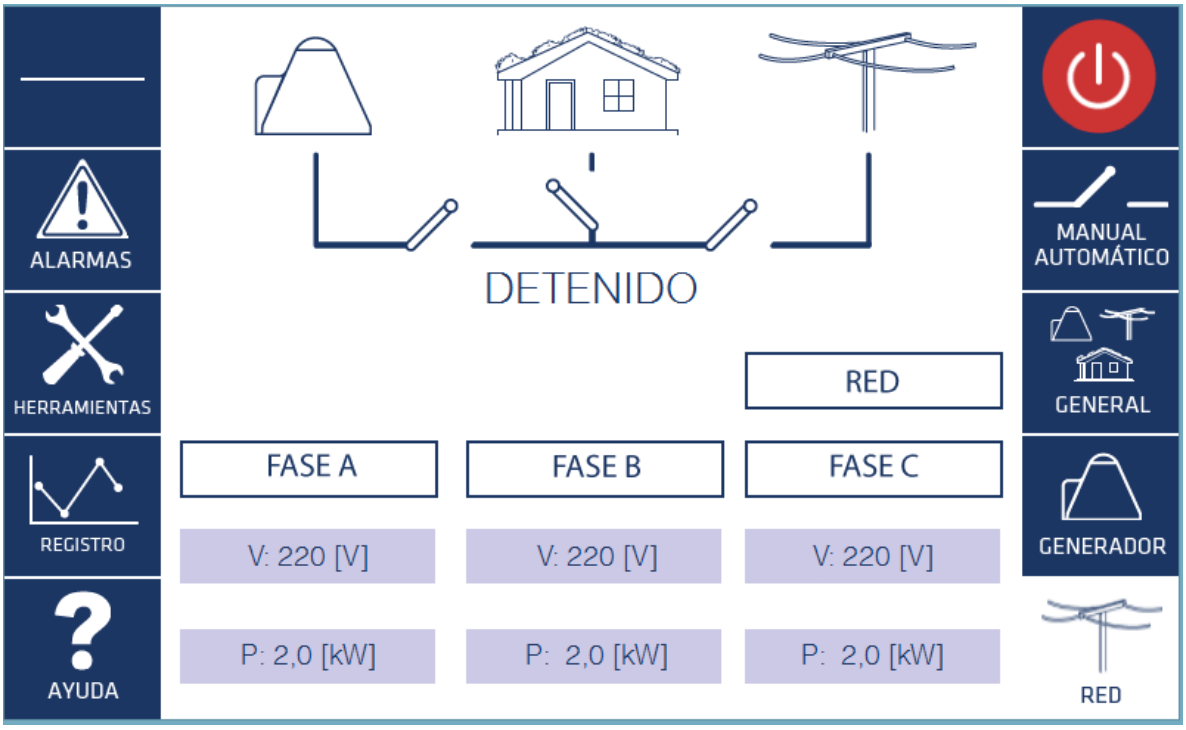


Figura 3-7: Pestaña de la red.

Tanto en la Figura 3-6 como en Figura 3-7 se observa la misma estructura que en la pestaña general, exceptuando que en estas toda la información pertenece a un solo componente del sistema.

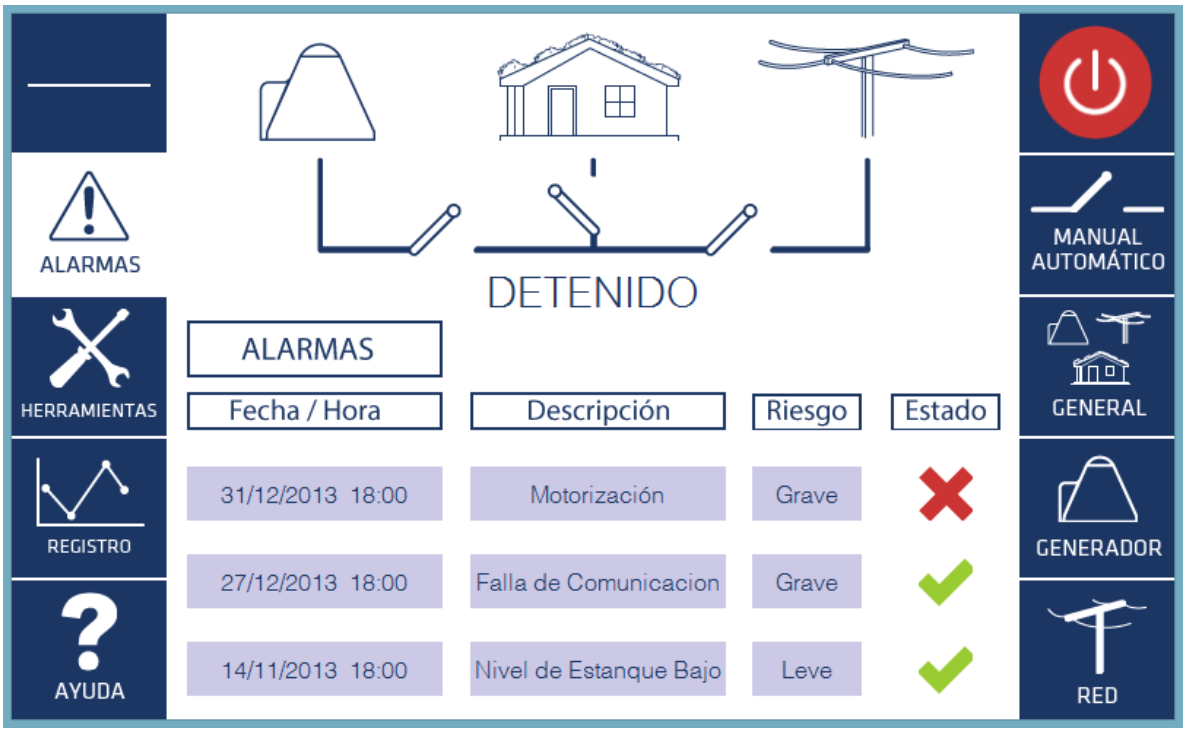


Figura 3-8: Pestaña de alarmas.

En la Figura 3-8 se muestran la pestaña de alarmas, esta se encuentra dividida en cuatro columnas, la primera corresponde a la fecha y hora en la cual se detectó una falla. En la segunda se indica la naturaleza de ésta, seguido del riesgo que representa la falla hacia el sistema. Finalmente, en la última columna se indica el estado de ésta, una "X" roja indica que no se encuentra resuelta, y un tick verde indica que la falla fue despejada correctamente.

Figura 3-9: Pestaña de herramientas

En la Figura 3-9 se observa la estructura de la pestaña de herramientas, ésta consta de cuatro botones: conexión wifi, carga de la batería del sistema, nivel de estanque, y comunicación local. Cada uno de estos botones debe permitir acceso a una sub-pestaña, en la cual, se observe el estado del componente. Sin embargo, estas sub-pestañas no forman parte de este trabajo, esto se debe principalmente que los sistemas de conexión wifi y batería del sistema no se encuentran en operación. Además, en el laboratorio de molina el caudal es simulado mediante una bomba, por lo tanto, no es necesario un sensor de nivel del estanque.

Figura 3-10: Pestaña de registro.

En la Figura 3-10 se muestra la pestaña de registro, ésta tiene como finalidad mostrar al usuario las tres últimas acciones que se han efectuado en el sistema, por ejemplo: transiciones, alarmas y ordenes ingresadas por el usuario. Junto con esta pestaña se implementó un archivo de texto plano el cual guarda todas las acciones realizadas desde el encendido de la máquina. De esta forma se puede construir una línea de tiempo con las operaciones realizadas por esta.

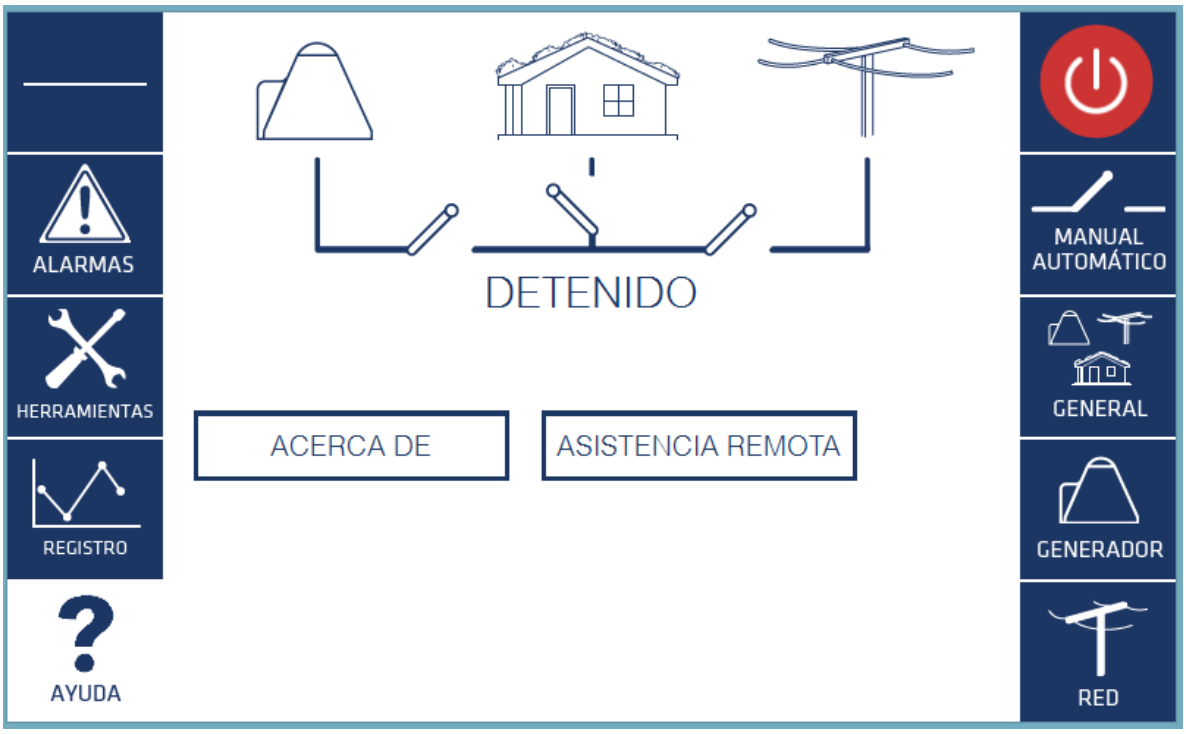


Figura 3-11: Pestaña de ayuda.

En la pestaña de ayuda, la cual se muestra en la Figura 3-11, se observan dos botones: el primero debería desplegar algún tipo de explicación del funcionamiento del sistema, el segundo debería llevar a la información de contacto del fabricante, estas dos acciones tampoco forman parte de este trabajo, esto se debe a que no está claro que tipo de información mostrar.

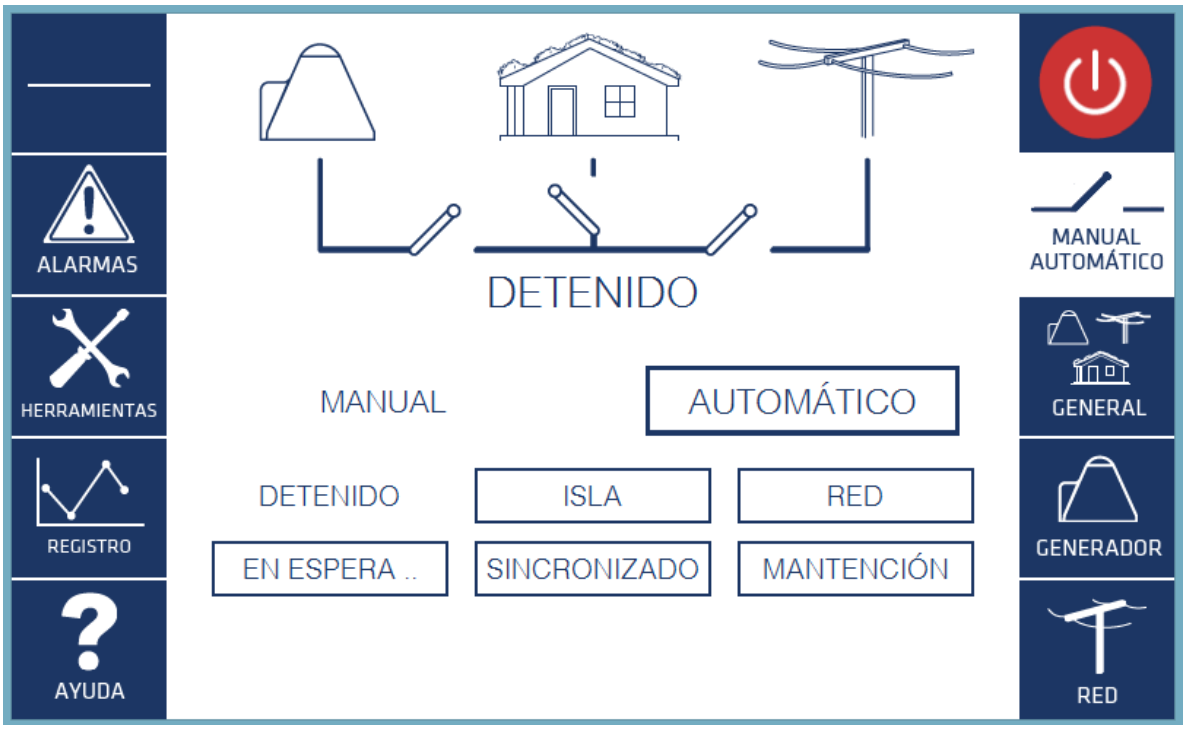


Figura 3-12: Pestaña de manual automático.

En la Figura 3-12 se observa la pestaña de manual automático. Esta pestaña permite al usuario modificar las consignas de operación del sistema. En particular los botones de manual y automático, determinarían la forma de operación del sistema. En el modo manual, será el usuario el encargado de realizar las transiciones presionando alguno de los botones de estados: Detenido, En Espera, Isla, Sincronizado, Red y Mantención. Por el contrario si se presiona el botón de modo automático, la HMI le envía una señal al DSP la cual le indica que este opere de forma autónoma. Ésta función todavía no se encuentra habilitada en el DSP.

Las imágenes mostradas en las figuras anteriores (Figura 3-4 a la Figura 3-12) forman parte de la tercera versión de la interfaz gráfica. Anteriormente se implementaron dos diseños los cuales se explicaran a continuación. El primero fue implementado por un equipo anterior y se puede observar en la Figura 3-13. Este fue construido en la plataforma QT y posteriormente montado en la SBC que se encuentra empotrada en la cúpula de la central.

Figura 3-13: HMI versión 1.

En la Figura 3-13 se observa la primera versión de la HMI, en particular se observa la pestaña de manual automático, en esta se permitía no solo modificar la forma y el estado de operación del sistema sino que también, la tensión y el factor de potencia de la unidad, esto se realizaba mediante barras deslizantes.

Figura 3-14: estructura general de la primera versión de la interfaz.

En la Figura 3-14 se observa la pestaña general de la primera versión de la HMI, en esta se observan las variables eléctricas de las tres componentes del sistema: generador, consumo y red. En cada componente se observa la tensión, frecuencia, corrientes de fase, potencia y finalmente el factor de potencia. Además, como se observa no existen pestañas de registro y herramientas. En la barra izquierda de la interfaz se observa el estado de la comunicación, el nivel de batería, horas de operación y alarmas. Mientras que en la barra derecha se encuentra el botón de encendido, y las pestañas: manual automático, general, generador, y red.

La segunda versión también, fue implementada utilizando la plataforma QT sin embargo esta fue montado sobre un computador utilizando un sistema operativo Linux. En particular, esta versión se agregó la pestaña de registro y se movieron los indicadores de estados de comunicación y batería a la parte superior de la interfaz. También, se eliminó el tiempo de operación y se agregó la hora y fecha actual. Además, se agregó una pestaña específica para la comunicación local del sistema (comunicación serial) y una pestaña de ayuda.

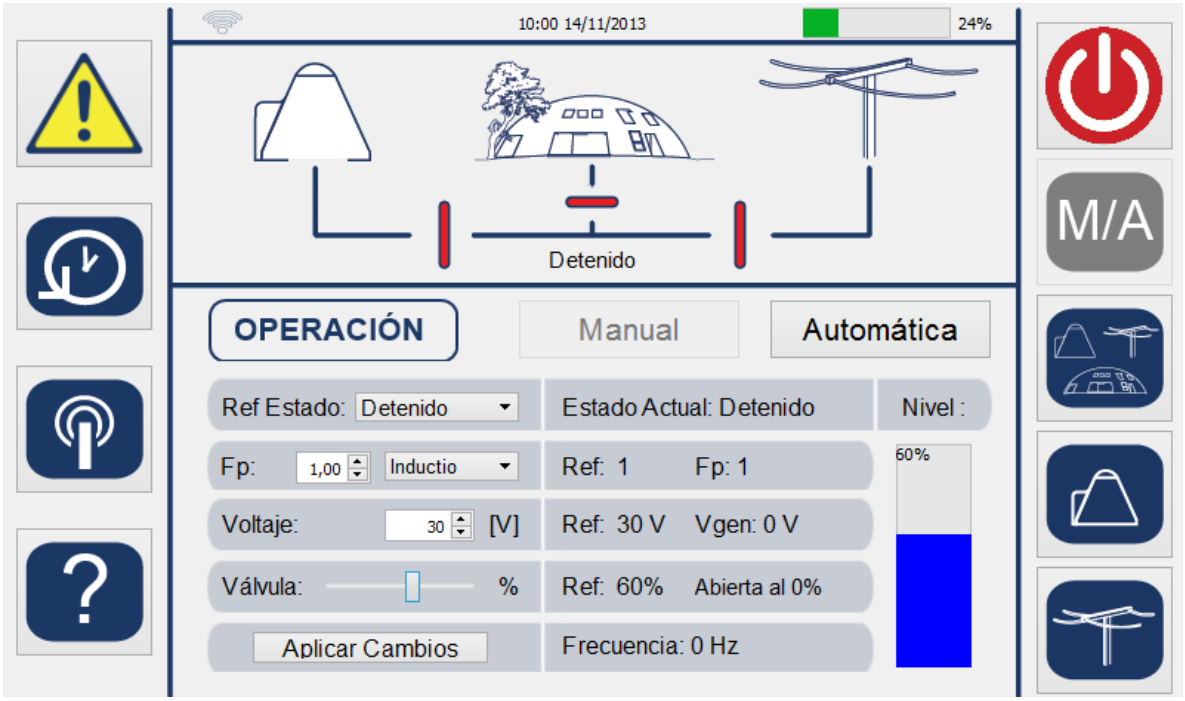


Figura 3-15: Pestaña de manual automático versión 2.

En la Figura 3-15 se observa la pestaña de manual automático de la HMI. En ésta se le permite al usuario modificar las consignas de forma y estado de operación, factor de potencia, tensión de generación y apertura de la válvula. Además se le informa del estado actual de operación, referencias y observaciones de tensión, factor de potencia y apertura de la válvula. Finalmente, también se muestra la frecuencia del generador y el nivel del estanque. Las consignas eran modificadas utilizando un menú desplegable o un cuadro de número.

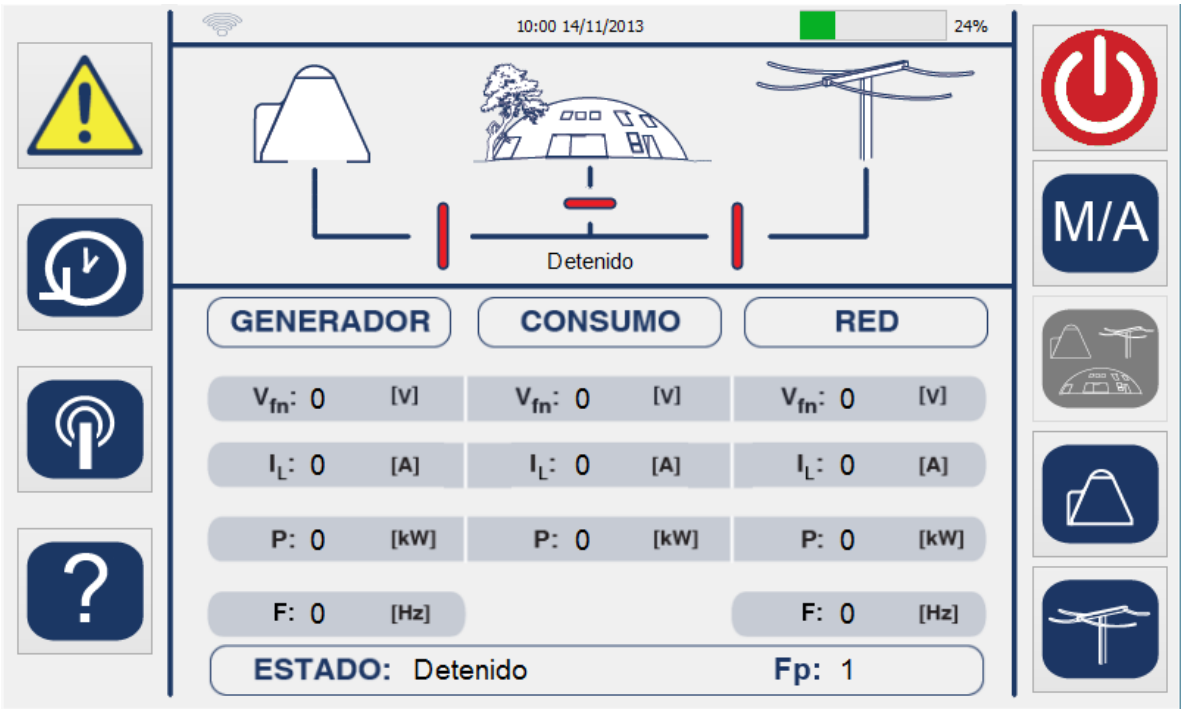


Figura 3-16: Pestaña general de la versión 2.

En la Figura 3-16 se observa la pestaña general, de la segunda versión de la interfaz. En esta se muestran las variables eléctricas: tensión rms fase neutro, corriente rms, potencia total y la frecuencia de cada uno de los componentes del sistema. Además, se le informa al usuario del estado operación del sistema, y el factor de potencia del generador.

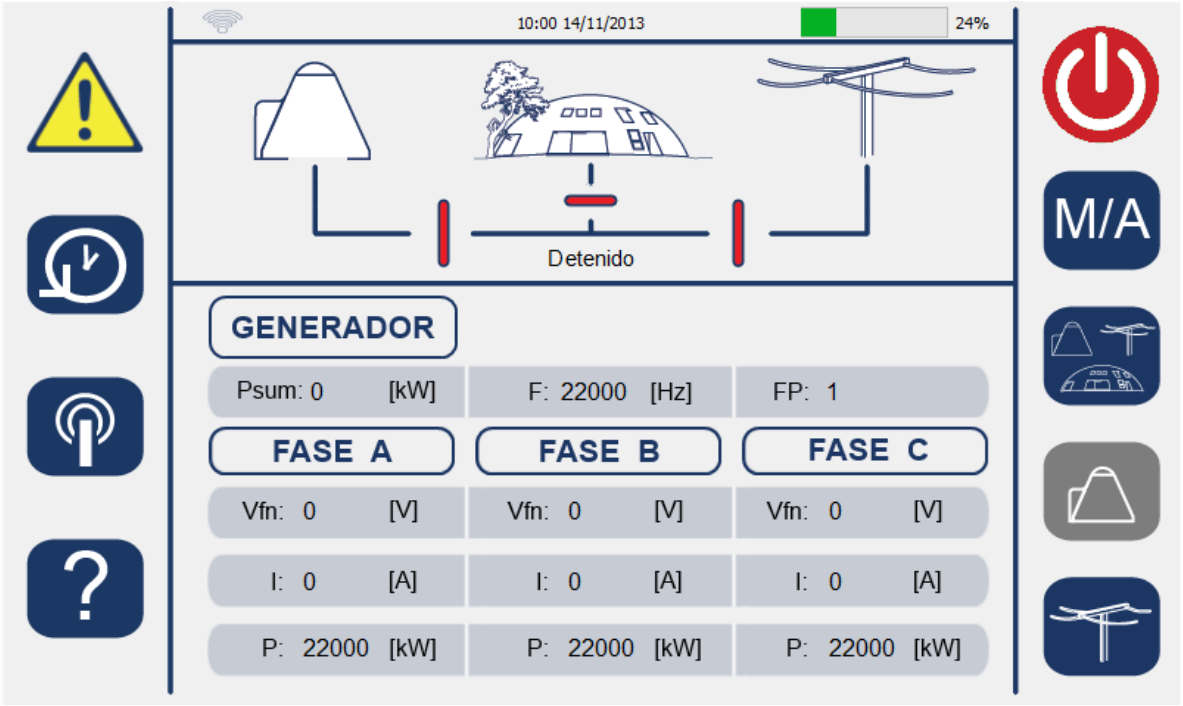


Figura 3-17: Pestaña detallada del generador.

En la Figura 3-17 se observa la pestaña de detalle de la generador. En ella se muestra: la tensión rms fase neutro de la unidad, las corrientes rms que circulan por cada línea, y la potencia que fluye por cada fase. Además en la parte superior se observa de izquierda a derecha: la potencia total generada, la frecuencia de operación y el factor de potencia del generador. La pestaña específica de la red posee la misma distribución grafica que la pestaña del generador y muestra el mismo tipo de información que ésta solo que referente a la red obviamente. Las demás pestañas no fueron implementadas, ya que durante el desarrollo de esta versión la gráfica fue actualizada y se cambió el sistema operativo del computador en el cual esta se estaba desarrollando.

Durante el cambio a la tercera versión se modificaron varios aspectos de la interfaz, el primero y más notorio fue la modificación visual que sufrió la interfaz. En versiones anteriores se utilizó predominantemente un fondo gris, con separadores entre las partes que conforman la interfaz y botones relativamente pequeños. En la nueva versión se agrandaron los botones, se eliminaron los separadores y se modificó el fondo de la interfaz, esto se realizó con el fin de hacerla más amigable con el usuario.

Además de las modificaciones gráficas, también ocurrieron modificaciones de fondo en esta transición. Por ejemplo se redujo la cantidad de información presentada al usuario. Esta modificación procura simplificar el diseño de la HMI, de modo que, el usuario no se vea abrumado por la cantidad de información presentada. Además el usuario puede no estar familiarizado con términos muy específicos como el factor de potencia. Bajo este marco se eliminó la de la interfaz los indicadores de, corriente, frecuencia y factor de potencia.

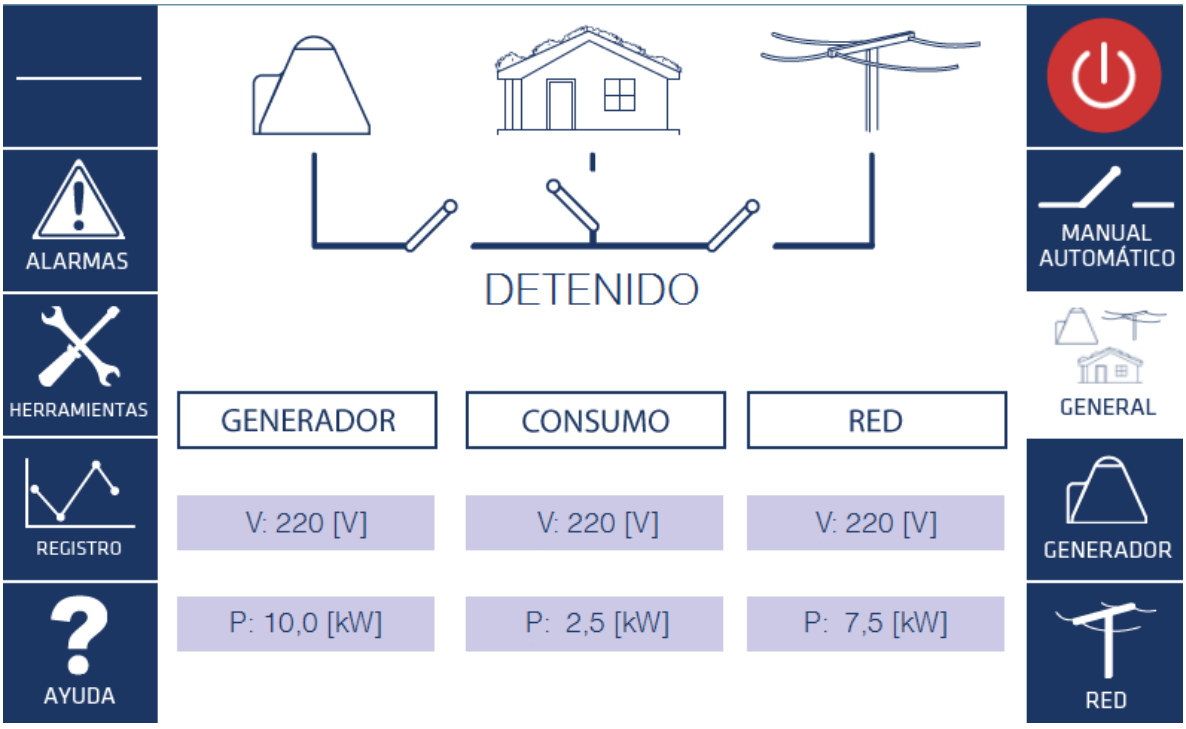


Figura 3-18: Pestaña general de la tercera versión.

En la Figura 3-18 se muestra la pestaña general de la tercera versión, en donde solo se presentan las tenciones y las potencias de cada componente. Además, también se observa el nuevo logo que representa el consumo local (anteriormente era una casa cúpula).

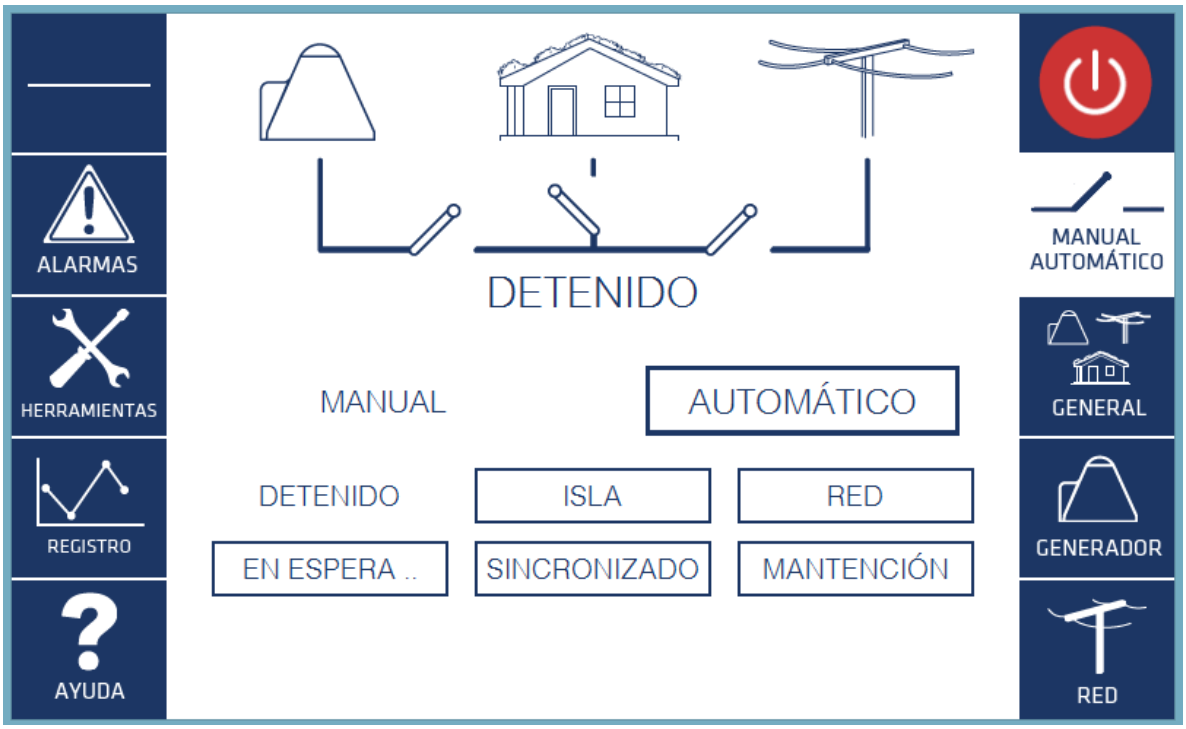


Figura 3-19: Pestaña de manual automático de la tercera versión.

Durante esta transición también, se modificó la pestaña de manual automático, en esta ahora solo se permite modificar la forma y estado de operación de la central. Además esto se realiza mediante botones, los cuales se encuentran habilitados o deshabilitados dependiendo del estado actual de operación.

Otra de las modificaciones realizadas durante esta etapa fue el cambio de sistema operativo del computador. en este caso la HMI fue implementada en Windows, sin embargo, el proyecto contempla la implementación de esta en una tablet con un sistema Android. Debido a esta razón se desarrolló una breve evaluación técnica económica para determinar el dispositivo en el cual se montara la interfaz.

Para este estudio se tuvieron en cuenta las siguientes restricciones: el dispositivo debe contar con o bien un puerto serial RS232 o un puerto USB OTG (On The Go), una pantalla de al menos 7” y un máximo de 10”, un procesador de almenas 1GHz con 512MB de memoria RAM.

Las características referentes al procesador buscan a evitar congelaciones o retardos en la interfaz, el tamaño de la pantalla fue determinado por el tamaño de la SBC anterior y el tamaño máximo de la cúpula de la central. Finalmente, se requiere que el puerto USB sea OTG ya que esta característica permite que alimente el conversor RS232-USB.

Se analizaron tres dispositivos: una tablet Lenovo A1000 y una tablet Lenovo S6000 y la SBC TS-TPC-8390-4710.

Tabla 3.4: Características de los dispositivos.

Características técnicas[2]			
Tipo	Tablet	Tablet	SBC
Modelo	Lenovo A1000	Lenovo S6000	TS-TPC-8390-4710
Procesador	1,2GHz	1,2GHz	1,066 GHz
Memoria RAM	1GB	1GB	512MB
Flash	4GB	16GB	512MB
Pantalla	7”	10,1”	7”
USB	Micro USB OTG	Micro USB OTG	4x USB
Puertos	Micro SD	Micro SD	I2C, SPI, 2x Ethernet, 4x RS232, 2x RS485, Micro SD
Wifi	Si	Si	No
Sistema Operativo	Android	Android	Linux Debían
Precio	CL$ 70.090	CL$ 159.990	USD$ 483 o CL$ 258.184
Link	Lenovo A1000 [11]	Lenovo S6000 [12]	TS-TPC-8390-4710 [13]

Se determina que la SBC será remplazada por la tablet Lenovo A1000. Esto es determinado por tres causas: la primera se debió a un ámbito físico. El soporte de pantalla en la unidad micro hidráulica es diseñado para una pantalla de 7”, por lo que, para instalar una Tablet de 10” habría que hacer modificaciones mayores en la cúpula de la unidad. Otro aspecto considerado fue la documentación. Hoy en día el sistema operativo Android es uno de los más utilizados en dispositivos móviles, por lo que existe una amplia documentación al respecto, sin embargo, si bien las SBC son utilizadas en el mercado, encontrar documentación de buena calidad para estas es un tanto más difícil. Finalmente, desde una perspectiva económica la Tablet Lenovo A1000 tiene un precio bastante inferior en comparación sus competidores.

Finalmente, una vez obtenida la tablet se procedió a probar la comunicación entre esta y el DSP, esto es realizado mediante una aplicación bajada desde Google Play. Se descargó la aplicación “Slick USB 2 Serial Demo” [14] desarrollada por la empresa “SlickDev”, la cual provee una biblioteca de USB Serial que permite utilizar conversores USB-Serial conectados a la tablet.

Por el lado del DSP se utiliza una rutina que envía un mensaje cada 300ms, el mensaje es el siguiente “ Timer isr!!! \n\r”. También posee la capacidad recibir mensajes, por ejemplo, si recibe el mensaje “Texas” entonces enviará una sola vez el mensaje “ Instruments! \n\r”. Ambos mensajes de salida tienen un lago de 16 caracteres. Este programa esta escrito con base a un

[2] Datos extraídos desde PC Factory

código de entrenamiento de la empresa Texas Instruments [7]. El código fuente se encuentra en el Apendice C.

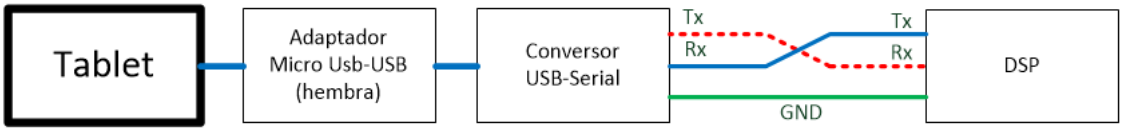


Figura 3-20: Esquema del montaje de prueba de comunicación.

Figura 3-21: Imagen de la prueba de comunicación.

En la Figura 3-20 se observa un esquema de las conexiones que se deben realizar para comunicar la tablet con el DSP. Partiendo desde la tablet se deben conectar en serie: un conversor micro-USB USB hembra, un adaptador USB serial, y finalmente tres cables hembra-hembra desde los pines: Rx, Tx y gnd del conversor, hacia los pines Tx, Rx, y gnd del DSP respectivamente. Por otra parte en la Figura 3-21 se muestra el montaje final utilizado, la tablet con la aplicación de demostración y el DSP.

3.4. Validación

Se deben validar dos sistemas, el primero corresponde al sistema de estados y transiciones que conforman el control supervisor de la unidad, mientras que el segundo corresponde a la HMI evaluada desde el punto de vista del usuario.

Para la primera evaluación se verifica el correcto funcionamiento del flujo del programa del DSP, vale decir que todas las interrupciones estén bien configuradas y por tanto se ejecuten periódicamente sin problemas, luego se procederá a verificar el funcionamiento de cada estado y las transiciones posibles para llegar a ellos. En cada estado se verifica lo siguiente.

1) Detenido:
   a) Control de campo desactivado (corriente de campo en cero).
   b) Tensión cero.
   c) Contactor generador abierto.
   d) Contactor red abierto.
   e) No sincronizado. (señal al control de frecuencia)
   f) Control de frecuencia desactivado. (señal al control de frecuencia)

2) En Espera:
   a) Control de campo activado en modo control de tensión.
   b) Tensión de generación $220V_{fn}$ con frecuencia de 50Hz
   c) Contactor generador abierto.
   d) Contactor red abierto.
   e) No sincronizado. (señal al control de frecuencia)
   f) Control de frecuencia activado. (a partir de 30 V)

3) Sincronizado:
   a) Control de campo modo factor de potencia.
   b) $\varphi = 0.45\ y\ \cos(\varphi) = 0.90$
   c) Contactor generador cerrado.
   d) Contactor red cerrado.
   e) Sincronizado. (señal al control de frecuencia)(Chopper al mínimo)
   f) Control de frecuencia activado.

4) Isla:
   a) Control de campo modo control de tensión.
   b) Tensión de generación 220 VV_{fn} y frecuencia de 50Hz
   c) Contactor generador cerrado.
   d) Contactor red abierto.
   e) No sincronizado. (señal al control de frecuencia)
   f) Control de frecuencia activado.

5) Red:
   a) Control de campo activado en modo control de tensión.
   b) Tensión de generación $220V_{fn}$ con frecuencia de 50Hz
   c) Contactor generador abierto.
   d) Contactor red cerrado.
   e) No sincronizado.
   f) Control de frecuencia activado. (a partir de 30 V)

6) Detenido:
   a) Control de campo desactivado.
   b) Tensión cero.
   c) Contactor generador abierto.
   d) Contactor red cerrado.
   e) No sincronizado.
   f) Control de frecuencia desactivado.

Además de los estados, también se debe probar las transiciones, en particular, éstas se deben ejecutar solo cuando se reciba la orden del usuario (en esta etapa desde el computador que observa el DSP y no desde la HMI). Además, éstas solo deben pasar al estado siguiente si las condiciones son adecuadas. En general, bajo este marco son las transiciones hacia isla, red, mantención y sincronizado las más restrictivas. En la primera se requiere que la tensión del generador sea de $220V_{fn}$ y la frecuencia de 50Hz, en la segunda y tercera se requiere que la red cumpla estas condiciones. Finalmente, la última requiere que las ondas de tensión tengan la misma amplitud $\pm 10V$, sus frecuencias sean prácticamente iguales, 0.1Hz de diferencia y se encuentren en fase, 0.03rad de error durante 200 muestras.

Además de mantenerse estable durante los estados y realizar las transiciones correctamente el sistema debe actuar correctamente ante una situación de falla, en particular en este caso se probará una falla de motorización de la máquina. Esto se realizará elevando el umbral de ésta de 100W a 1000W, de modo que al disminuir el caudal se traspase este umbral y el sistema transite correctamente a estado Isla.

Por otro lado la validación de la HMI requiere que la estructura de esta facilite la creación del modelo mental que se genera en el usuario. Para medir esto se utiliza un test que ofrece la guía para la evaluación experta [9]. Este test formula las siguientes preguntas.

1) ¿Aparece la navegación en un lugar preminente, donde se vea fácilmente?
2) ¿Los enlaces hechos de imágenes tienen su atributo escrito?
3) ¿Tiene enlaces rotos? Si los tiene quítelos.
4) ¿Posee un mapa o un buscador con el cual presentar una alternativa de navegación?
5) ¿Se mantiene una navegación consistente y coherente?
6) ¿existen elementos que le permitan al usuario saber siempre en donde se encuentra y que le permitan volver atrás?
7) ¿Indican los enlaces claramente hacia dónde apuntan? ¿Está claro lo que el usuario encontrara detrás de cada uno de ellos?
8) ¿La redacción es corta y precisa? Evite textos extensos.
9) ¿Ocupa colores amigables y que concuerden con los objetivos y propósitos de la aplicación?
10) ¿Hay espacios blancos (libres) entre el contenido, para descansar la vista? ¿O existe una saturación de contenido que desconcierta al usuario?
11) ¿Se han previsto respuestas del sistema frente a interacciones con el usuario? (ej: mensajes de confirmación, diálogos, etc.)
12) ¿Puede el usuario ponerse en contacto para hacer sugerencias o comentarios?

Finalmente, se aplican pruebas de comunicación las cuales consisten en comparar la información mostrada por CCS y la interfaz. Finalmente, se procede a probar todas las transiciones y estados utilizando solamente la HMI.

Capítulo 4. Pruebas Experimentales

4.1. Descripción General

En esta sección se presentan los resultados obtenidos a partir de las pruebas experimentales de ambos sistemas en el laboratorio de Molina. Estas pruebas consisten en la aplicación del test de validación de la estructura de control supervisor propuesta en el capítulo anterior. Durante éstas se procede a llevar el sistema a cada uno de los distintos estados de operación posibles, utilizando las transiciones adecuadas, disparadas por el usuario desde el software CCS.

También, se realizara pruebas de comunicación entre la HMI y el DSP, verificando que los mensajes lleguen a ambos dispositivos sin problemas, y desencadenaran las acciones adecuadas. Nuevamente se utiliza CCS para monitorear el DSP y la consola provista por QT para la HMI. En este ámbito se verifica que las variables mostradas en la interfaz se muestren correctamente y no presenten grandes oscilaciones. Finalmente, se procede a ejecutar el test de validación gráfico propuesto por la guía de evaluación experta [9].

Una vez obtenidos estos resultados se procede a realizar un análisis, del comportamiento del sistema durante estas pruebas. En particular, en el caso de que se hayan presentado problemas durante las pruebas se procede a identificar el problema y poner una solución a éste para posteriormente realizar nuevamente la prueba necesaria para validar el sistema.

4.2. Resultados Obtenidos

Los primeros resultados se obtienen en el laboratorio de Molina. Durante estas pruebas se procede a probar los distintos estados de operación del sistema. Para lograr esto se conecta el DSP de la unidad con un computador utilizando un JTag, el que permite monitorear en tiempo real las variables internas del DSP. Mediante este dispositivo y CCS se logró desencadenar la las distintas transiciones en el DSP.

Al encender el DSP la central automáticamente entra en el estado detenido. Durante este estado se mide la corriente que circula por el campo de la máquina, comprobando que durante éste no existía circulación de corriente. Además, se observa que ambos contactores permanezcan abiertos. Luego se procede a transitar al estado en espera. Esta transición no presenta ningún problema, realizándose de manera instantánea.

En el estado en espera se procede a aumentar el flujo de agua, y posteriormente a verificar la tensión en bornes y frecuencia de operación de la unidad, comprobando que la tensión aumenta de manera proporcional a la frecuencia. Una vez que la máquina alcanza los 50Hz, el control de frecuencia se enciende y la máquina se mantiene alrededor de los $220V_{fn}$ y 50Hz. Durante este estado nunca se supera el límite superior de corriente de campo 2A, además no se activa ninguno de los contactores.

El siguiente paso consiste en provocar una transición a estado isla. Durante esta primera transición la carga local era nula. Se comprueba que ésta se realiza sin problemas, observando que el contactor del generador se cierra sin problemas y la unidad permanece operando de

forma normal. Luego se procede a agregar una carga al sistema de aproximadamente 1kW por fase. Se provoca un descenso en la frecuencia, por lo que, el caudal aumenta.

Luego se procede a migrar al estado en Espera. Esta migración se realiza sin problemas ya que al abrir el contactor del generador la unidad se estabiliza en torno a los $220V_{fn}$ y 50Hz.

Estas transiciones son probadas varias veces antes de realizar una sincronización. Durante estas pruebas se detecta que en algunos casos, la apertura del contactor del generador provocaba la pérdida de comunicación entre el DSP y el computador, instancias en las cuales se debió des-energizar el sistema y cortar el flujo de agua. También en algunos casos se presenta una pérdida total del campo del generador y la caída del DSP.

Una vez que se comprueba que el estado isla funcionaba correctamente se procede a probar la transición de sincronización. Durante los primeros intentos el sistema se falla, ya que los automáticos del sistema se activan provocando que la central pase a operar en vacío y por tanto el generador se acelera. Debido a esta reacción se procede a conectar un sincronoscopio entre la red y el generador. Se observa que al momento de producirse el cierre del contactor entre la red y el generador, las ondas de tensión no tienen la misma frecuencia y fase. Con base a esto se opta por aumentar la cantidad de ciclos en los cuales se deben mantener las condiciones de sincronización. En particular la condición de fase se debe mantener durante las 200 últimas muestras. Una vez modificadas las condiciones para la sincronización, se logra realizar la transición sin problemas y suavemente.

Una vez sincronizados se verifica que la máquina esté inyectando potencia al sistema. También se comprueba que la corriente de campo esté dentro de los límites establecidos, y luego se procede a probar las transiciones hacia modo isla y en espera. Nuevamente se presentan problemas de comunicación y pérdida de la corriente de campo al transitar hacia modo en espera.

Finalmente, se prueban las transiciones hacia los estados en mantención y solo red. Durante éstas no se presentan problemas, exceptuando las transiciones desde el estado isla o sincronizado hacia el estado solo red. Durante estas dos transiciones se presentan ocasiones en las que se pierde la comunicación y la corriente de campo del generador. Por otro lado la operación en estos estados no presenta problemas.

Eventualmente, el problema de comunicación y pérdida de corriente de campo es solucionado mediante el cambio de conexión de los contactores. Anteriormente éstos se encuentran conectados a las fases del generador. Al cambiar la alimentación a la red este problema deja de manifestarse.

Luego de realizar las pruebas de funcionamiento del sistema de transiciones y estados de operación, se procede a realizar una simulación de una falla de motorización. Para esto se sube el umbral de activación de la falla a 1kW y luego se disminuye el caudal, hecho que provoca que el sistema migrara al estado isla. Se verifica de esta forma el funcionamiento del sistema ante esta falla.

El siguiente paso fue probar el funcionamiento de la HMI en el sistema. Como se presenta en secciones anteriores se pasa por tres versiones de la HMI. La primera desarrollada por el equipo anterior y que fue montada sobre la SBC. La segunda fue montada sobre un computador con Linux y finalmente la tercera sobre un computador con Windows. Durante la prueba de la

primera versión se detectan varios problemas. El más grave de todos corresponde a la congelación de la interfaz y retardos presentados ante interacciones con el usuario. En particular al presionar un botón, la interfaz se "congelaba" durante un par de segundos. Otro aspecto era la interfaz misma, la que permite modificar consignas mediante botones y barras deslizantes. En particular, estas últimas hacen difícil ingresar una consigna ya que presentan un bajo tiempo de respuesta y son demasiado sensibles al cambio de posición de la barra.

La segunda versión de la HMI logra establecer comunicación con el DSP y por ende se logra observar las variables eléctricas del sistema e ingresar cambios de operación. Sin embargo presenta los siguientes problemas:

- Retardos en la visualización de datos. Para determinar esto se compara los datos presentados en la interfaz con los datos presentados en Code Composer, donde se determina que existía un retardo en la visualización de aproximadamente 7 segundos. Esto es evidente durante los cambios de estado ya que los contactores se activan varios segundos antes del cambio de estado en la interfaz.

- Congelamientos e interferencias de la interfaz. En esta versión la interfaz presentan retardos e incluso, omisión en los comandos ingresados por el usuario de la interfaz, en particular este comportamiento suele ocurrir cuando la central se encuentra operando en estado isla o en espera.

- Inestabilidad. Otro problema observado son los reiterados colapsos de la aplicación al momento de transitar entre estados. Estos colapsos suelen ocurrir durante los rechazos de carga, y son acompañados de un colapso del programa del DSP. Los problemas relacionados al rechazo de carga son solucionados al cambiar la alimentación de los contactores.

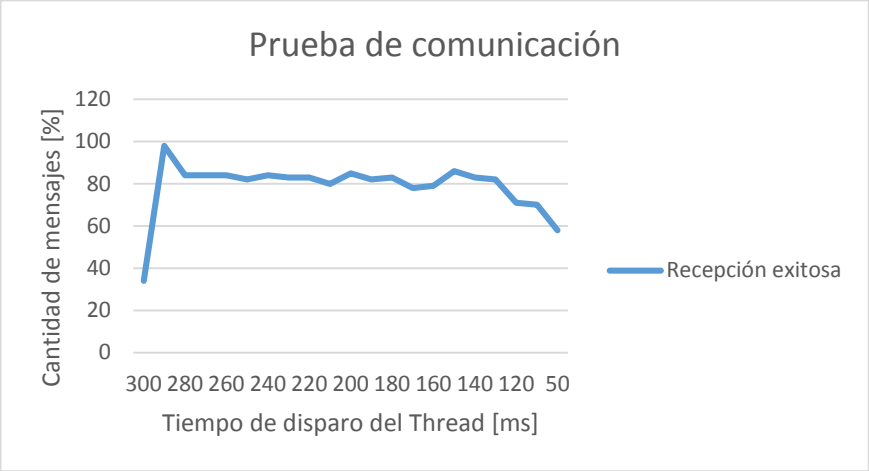
Durante estas pruebas el sistema de comunicación de esta interfaz es asociado al hilo principal de ésta, el cual se encarga de las funciones gráficas de la aplicación. Una vez obtenido estos resultados se procede a actualizar la aplicación. En particular, se mueve el sistema de comunicación hacia un thread independiente disparado por un reloj. Esta actualización elimina los retardos de visualización de datos y congelamientos. Sin embargo, el problema de interferencias sigue presentándose y aumenta el número de colapsos globales de la aplicación, es decir, esta se hace más inestable.

En la tercera versión de la interfaz se migra el sistema a una plataforma de Windows. Además, se introducen sistemas de bloqueo de variables, el cual mejora de forma significativa la estabilidad de la aplicación. Sin embargo, se vuelven a presentar interferencias.

Otro aspecto observado fue la estabilidad de las variables mostradas. En general estas permanecen relativamente estables durante la mayoría de los estados de operación. Sin embargo, durante el estado sincronizado las potencias totales del generador, la red y el consumo presentan oscilaciones de alrededor de 300W y en particular en el caso del consumo de hasta 600W.

Durante todas estas pruebas se presentan problemas de pérdida de comunicación, en particular durante los rechazos de carga se observa que desconectar el adaptador serial y volverlo a conectar soluciona el problema.

Otra prueba realizada fue la velocidad de disparo del thread de comunicación. En esta se compara la cantidad de mensajes recibidos correctamente con la cantidad total de mensajes enviados (1000 mensajes), se obtuvieron los siguientes resultados.



Gráficos 4.1: Prueba de disparo del thread de comunicación.

En el Gráficos 4.1 se observa que para una tasa de disparo de 290ms se reciben exitosamente el 98% de los mensajes enviados.

Luego de estas pruebas se le aplica el test de evaluación [9] a las dos últimas versiones de la interfaz. Cabe notar que este test fue realizado a un número muy limitado de personas, por lo que no representa bajo ningún concepto la validación final de la HMI. Con este test se busca obtener una idea del estado de avance de la interfaz, los resultados fueron los siguientes.

Segunda versión:

1) ¿Aparece la navegación en un lugar preminente, donde se vea fácilmente?
   Si. Efectivamente la navegación aparece en los costados de la aplicación en todo momento

2) ¿Los enlaces hechos de imágenes tienen su atributo escrito?
   No. Aún más, algunas de estas imágenes no reflejan correctamente el contenido que contienen.

3) ¿Tiene enlaces rotos? Si los tiene quítelos.
   Si, la pestaña de ayuda, el botón de encendido, la pestaña de comunicaciones, registro, alertas.

4) ¿Posee un mapa o un buscador con el cual presentar una alternativa de navegación?
   No.

5) ¿Se mantiene una navegación consistente y coherente?
   Si. Cada botón lleva a un contenido específico de un tema.

6) ¿Existen elementos que le permitan al usuario saber siempre en donde se encuentra y que le permitan volver atrás?
   Si, el botón de la pestaña seleccionada se oscurece cuando esta se está mostrando, sin embargo, la relación entre el oscurecimiento del enlace a la pestaña y la pestaña no es inmediato y debe ser mejorado.

7) ¿Indican los enlaces claramente hacia dónde apuntan? ¿Está claro lo que el usuario encontrará detrás de cada uno de ellos?
   No, tanto los botones de comunicación, manual/automático y registro no representan correctamente su contenido.

8) ¿La redacción es corta y precisa? Evite textos extensos.
   Es demasiado corta, se requieren de más textos explicativos, en especial en los botones.

9) ¿Ocupa colores amigables y que concuerden con los objetivos y propósitos de la aplicación?
   En general hubo una buena aceptación en relación al aspecto de la aplicación, eso so hubieron comentarios de agregar colores verdes y rojos para resaltar la información importante.

10) ¿Hay espacios blancos (libres) entre el contenido, para descansar la vista? ¿O existe una saturación de contenido que desconcierta al usuario?
    No existen espacios en blanco, y si se percibe una recarga de contenido. Además la operación de la central es muy compleja.

11) ¿Se han previsto respuestas del sistema frente a interacciones con el usuario? (ej.: mensajes de confirmación, diálogos, etc.)
    No existen acciones de confirmación ni diálogos con el usuario.

12) ¿Puede el usuario ponerse en contacto para hacer sugerencias o comentarios?
    No, se tiene contemplado pero, todavía no se ha implementado.

Tercera versión:

1) ¿Aparece la navegación en un lugar preminente, donde se vea fácilmente?
   Si. Nuevamente la navegación fue colocada en los bordes de la aplicación.

2) ¿Los enlaces hechos de imágenes tienen su atributo escrito?
   Sí, pero, es insuficiente, falta una breve explicación de su función. En particular se sugirió cambiar los textos de los botones de Manual/Automaático, Registro y

Herramientas. Los más sugeridos fueron: Cambios de operación, Historial de operaciones, y Ajustes respectivamente.

3) ¿Tiene enlaces rotos? Si los tiene quítelos.
   Si, la pestaña de ayuda, el botón de encendido, la pestaña de comunicaciones, registro, alertas.

4) ¿Posee un mapa o un buscador con el cual presentar una alternativa de navegación?
   No.

5) ¿Se mantiene una navegación consistente y coherente?
   En la mayoría de los casos sí, en particular la pestaña de herramientas posee contenido que no debería estar en esta como el nivel del estanque y el estado de carga de las baterías.

6) ¿Existen elementos que le permitan al usuario saber siempre en donde se encuentra y que le permitan volver atrás?
   Si bien se avanzó en este aspecto aún se requiere una forma más fácil de identificar la pestaña que se está observando.

7) ¿Indican los enlaces claramente hacia dónde apuntan? ¿Está claro lo que el usuario encontrara detrás de cada uno de ellos?
   No: el enlace de Herramientas lleva a algunas mediciones y el enlace de manual/automático debería tener otro nombre.

8) ¿La redacción es corta y precisa? Evite textos extensos.
   Nuevamente, muy poco texto, faltan pequeñas explicaciones para los botones en las barras de navegación.

9) ¿Ocupa colores amigables y que concuerden con los objetivos y propósitos de la aplicación?
   Se ve amigable pero, aún se pueden explotar más los colores, en particular el verde y el rojo para resaltar información.

10) ¿Hay espacios blancos (libres) entre el contenido, para descansar la vista? ¿O existe una saturación de contenido que desconcierta al usuario?
    Sí, no hay una recarga de información y las operaciones posibles de realizar son bastante más sencillas.

11) ¿Se han previsto respuestas del sistema frente a interacciones con el usuario? (ej: mensajes de confirmación, diálogos, etc.)
    No, faltan diálogos de confirmación, y pequeños mensajes ante las operaciones de usuario.

12) ¿Puede el usuario ponerse en contacto para hacer sugerencias o comentarios?
No se ha implementado todavía.

Finalmente, referente a la comunicación entre el DSP y la tablet se obtienen los siguientes resultados. La prueba de conexión muestra que es posible conectar ambos dispositivos y establece un canal de comunicación serial mediante un conversor USB-Serial. Se logra recibir información desde el DSP sin problema alguno y además, los mensajes se muestran en pantalla conforme al tiempo de envío. La aplicación es capaz de enviar el mensaje "Texas" a lo que el DSP respondió "Instruments", probando exitosamente la comunicación en ambos sentidos.

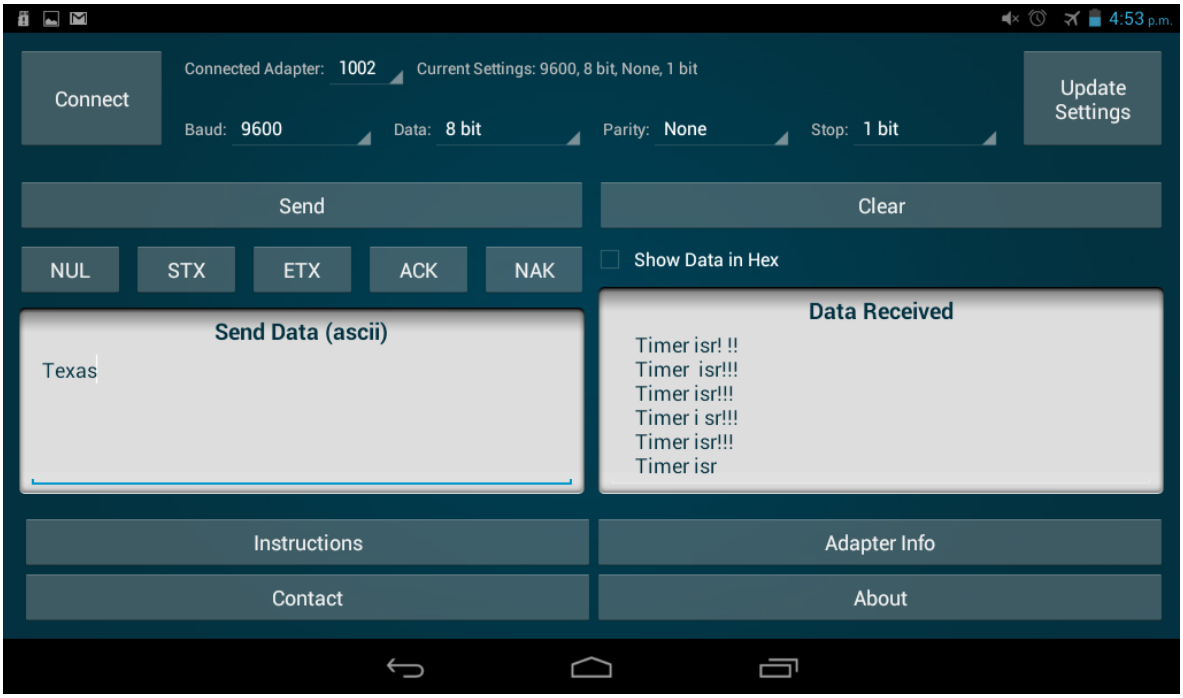


Figura 4-1: Recepción del mensaje desde el extremo de la Tablet.

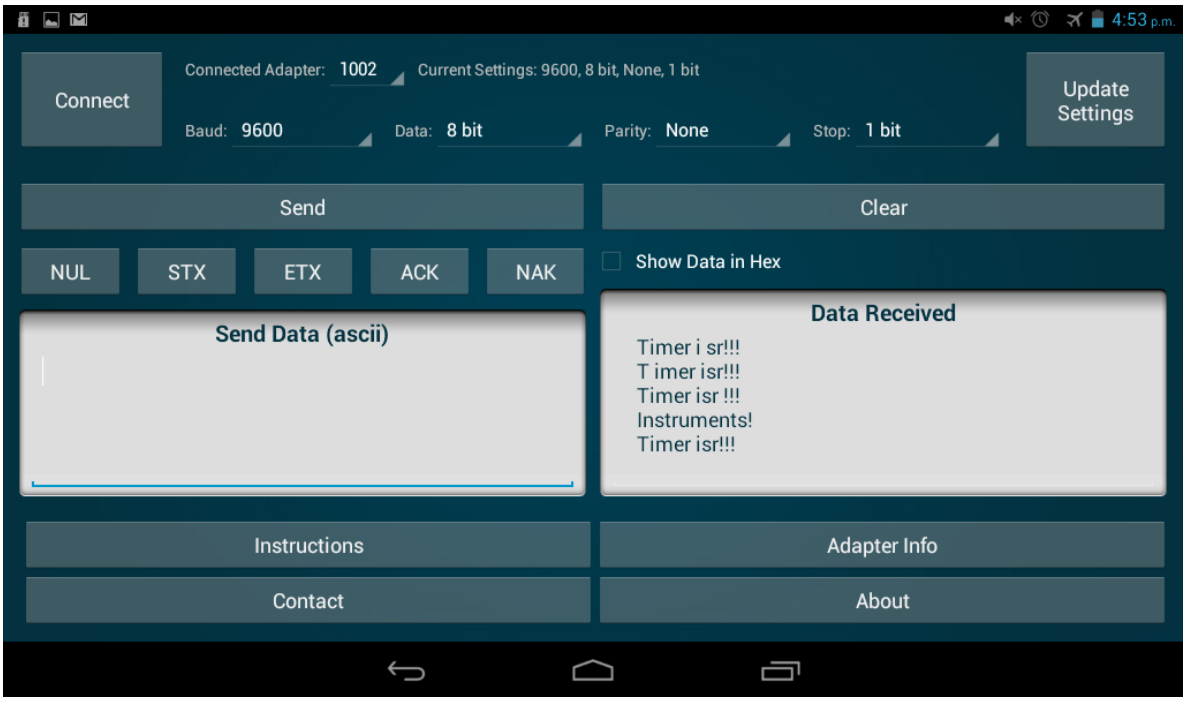


Figura 4-2: Recepción y respuesta desde el DSP.

En la Figura 4-1 se muestra en el lado izquierdo el mensaje a ser enviado desde la tablet y en el lado derecho la recepción de los mensajes enviados desde el DSP. Por otro lado en la Figura 4-2 se muestra en el lado derecho la recepción del mensaje “Instruments!” lo cual verifica el correcto funcionamiento del canal de comunicación.

4.3. Análisis de Resultados

El esquema de estados y transiciones propuesto en este trabajo produce resultados positivos, si bien en un principio la unidad presenta algunos problemas durante los tránsitos entre estados tales como las fallas de sincronización en una primera instancia y posteriormente los colapsos de la operación del DSP, estos problemas son solucionados.

En particular, el problema de sincronización se debía a un error en el resultado del test de condiciones de sincronización, en particular a un falso positivo. Al aumentar el tiempo en que estas condiciones se deben mantener se redujo la tasa de sincronizaciones fallidas, al punto en que no se han vuelto a producir. Un aumento del tiempo le confiere suavidad a la operación. Esto se debe a que las condiciones en ambos componentes sean realmente similares, ya que si el tiempo es muy pequeño puede ocurrir que el generador se acerque momentáneamente a las condiciones de la red, provocando el cierre del contactor, y dado que el cierre no es instantáneo, puede ocurrir que las condiciones del sistema hayan cambiado.

Si las condiciones en ambos componentes del sistema son muy distintas se producirá un cambio de velocidad muy brusco en la máquina. Este cambio está acompañado por pick de corriente, que provoca la activación de los automáticos del sistema. La activación del sistema de protección, lleva a la salida del control de frecuencia y la pérdida total de la carga provocando que la máquina se acelere. Por otro lado si el tiempo en el cual se requiere que las condiciones

de sincronización es muy extenso, la transición tomará demasiado tiempo en realizarse. Esto se debe a que sistema no cuenta con un control de la fase, por lo que se debe esperar a que estas condiciones se den naturalmente. Finalmente, se determina de manera experimental que la condición de frecuencia se debe mantener durante 1000 muestras consecutivas, y la condición de fase durante las 200 últimas.

Por otra parte los colapsos del DSP se producen por un problema con los contactores del sistema. La apertura de uno de estos provocaba la caída del sistema. Este problema fue solucionado cambiando la alimentación de los contactores desde el generador a la red. Desafortunadamente este tipo de solución sólo puede ser temporal. Esto se debe a que el objetivo final contempla la operación de la unidad en situaciones de ausencia de red.

Además, los problemas de colapso del DSP deben ser solucionados por el WatchDog. Este sistema consta de dos llaves antibloqueo, las cuales no son activadas cuando se produce un colapso, evento que debería reiniciar el dispositivo. Dado que el sistema en esta etapa no cuenta con una forma de almacenamiento de datos es imposible recuperar el estado previo de operación. Por lo que se requeriría una forma de almacenar información de forma indefinida en el sistema.

Durante la operación del sistema se observan pérdidas de comunicación con la HMI, en particular estas se producen durante la operación en modo isla o en espera. Estos dos estados de operación solo poseen un componente en común, el cual no está presente en los demás estados de operación, el control de frecuencia. Este se encarga de mantener la frecuencia del sistema en torno a los 50Hz, objetivo que se logra a través de el consumo de energía en una resistencia de desahogo. Para esto el sistema se vale de un rectificador y un chopper. Estos elementos generan una gran cantidad de ruidos en el sistema, los cuales pueden afectar a los distintos dispositivos que lo componen.

Si además, se agrega un evento de rechazo de carga, en el cual se genera la apertura de uno de los contactores, y se requiere que toda la potencia que antes era consumida por la carga local sea transferida al sistema de control de frecuencia. Se obtiene que instantáneamente el sistema se ve severamente perturbado. Si consideramos que la electrónica de control se encuentra repartida en varias placas conectadas a través de cables de teléfono, el colapso del DSP y la comunicación es casi inevitable. Para volver a levantar el canal de comunicación entre el DSP y la HMI se require de energizar el conversor serial-USB entre estos dispositivos. Esta acción apunta a la existencia de algún tipo de protección dentro de estos conversores la cual se debe tener en cuenta en posteriores diseños.

En lo que atañe concretamente al uso de la HMI se puede extraer que durante las pruebas de la primera versión se observa que ésta tendía a presentar errores de congelación de la interfaz y retardos frente a la interacción con el usuario, todo esto mientras el sistema en general se encontraba apagado. Este comportamiento apunta a las características técnicas de la HMI, en particular la SBC posee un procesador de 200MHz y 128MB de RAM. Estas características son insuficientes.

Esto se debe a que el programa de la interfaz posee dos componentes. Por un lado un programa que se encarga de la recepción y procesamiento de datos desde el puerto serial, mientras que el segundo se encarga del aspecto gráfico de ésta, ambos unidos a través de un

canal de comunicación interno. Sin embargo, dado que la interfaz fue programada en QT, se requerirá ejecutar rutinas específicas de este programa para mantener su operación.

Además, de estos programas se debe agregar el soporte gráfico interno de la SBC (Server X) sin el cual, ninguna aplicación gráfica podrá ser ejecutada. Todos estos programas funcionando de manera paralela provocan que el procesador de la SBC se vea sobrepasado, y por tanto la interfaz no responda adecuadamente. Por lo que si se quiere implementar una aplicación en la interfaz, ésta se debe hacer en el lenguaje nativo de ésta.

Durante las pruebas de la segunda versión se observan: retardos en la visualización de datos, congelaciones y problemas de usabilidad. Los retardos y congelaciones se deben principalmente a la estructura del programa de la interfaz, durante esta versión, la comunicación serial y el manejo gráfico de la aplicación estaban unidos en un mismo hilo de ejecución. Por lo anterior, al ejecutarse la rutina de lectura del puerto serial, la parte gráfica quedaba bloqueada hasta que se completara la lectura de un menaje correcto. Debido a esta razón se decide separar los módulos de comunicación y manejo gráfico de la interfaz.

Por otro lado los problemas de usabilidad poseen una fuente totalmente distinta. Durante esta etapa del proyecto se observa una gran cantidad de interferencia en los distintos periféricos de los dispositivos conectados con el sistema o en el entorno de este, en particular tanto el ratón como el panel táctil de notebook en el cual la interfaz fue implementada presentan problemas de funcionamiento, dificultando el uso de la interfaz. Este problema se genera por la interferencia electromagnética generada por el sistema. Este efecto puede volver a presentarse en la versión final de la HMI ya que ésta contaría con una pantalla táctil la cual es susceptible a este tipo de interferencias.

En la tercera versión se implementa el sistema de hilos de ejecución paralelos para la comunicación y la parte gráfica de la interfaz. Este sistema corrige los problemas de retardos y congelaciones presentados en la versión anterior de la HMI. Sin embargo, luego de la implementación de este se observa un aumento circunstancial de la inestabilidad de la aplicación. Esto se debe a que durante ésta se producen choques entre los hilos del módulo gráfico y de comunicación. Estos choques se producen cuando ambos hilos intentan acceder a la misma variable produciéndose una falla. Finalmente, este problema es solucionado mediante Mutex, los cuales actúan como un semáforo permitiendo el acceso de solo uno de los hilos a cada variable.

Además de este problema, la interfaz presenta problemas de la estabilidad de las potencias durante el estado sincronizado. Para solucionar este problema se incluyen filtros pasa bajos para las variables mostradas durante este estado de operación. Estas oscilaciones pueden explicarse por oscilaciones del control de factor de potencia. De todas estas variables es la potencia total del consumo la que presenta mayor variabilidad. Esto se debe a que esta potencia está siendo calculada utilizando las potencias del generador y la red, las cuales ya poseen oscilaciones.

Finalmente, se realiza una prueba para determinar la frecuencia de disparo óptima para el thread que maneja la comunicación serial. Se encuentra que ésta corresponde a 290ms con lo cual se debería comenzar a observar el puerto serial justo antes del inicio de transmisión en el DSP, ya que un nuevo thread solo comienza una vez que el su predecesor ha muerto. Por lo

anterior se produce una suerte de sincronización entre el envío de mensajes desde el DSP y el thread de lectura en la HMI.

A partir del test de evaluación propuesto se puede concluir lo siguiente: la navegación en una interfaz gráfica como la propuesta es un punto vital dentro del diseño de una aplicación, ya que ésta debe ser lo suficientemente clara para que el usuario entienda de forma intuitiva el funcionamiento de la interfaz misma. En particular, todas las versiones poseen la virtud de siempre mantener el menú de navegación a la vista del usuario. Sin embargo, las primeras versiones carecen de un texto que explique el contenido de cada una de las pestañas de la interfaz y la última, si bien presenta un texto, parece ser insuficiente para representar correctamente el contenido de ésta.

Además, no todas las imágenes representan correctamente el contenido al que apuntan, en particular, las pestañas de Comunicación y Manual/Automático en la segunda versión, y las pestañas de Herramientas y Manual/Automático en la tercera. Esta última se encuentra representada por un contactor con el cual se busca dar la idea de que en esta pestaña se puede modificar el estado de las conexiones de este sistema. Sin embargo, un contactor es un dispositivo con el cual el común de las personas no está familiarizado. Además, el nombre de esta pestaña no representa correctamente la función de realizar transiciones en el sistema.

Por otro lado la pestaña de herramienta no solo presenta un problema de representación del contenido, sino que también, posee un error de consistencia. La palabra herramientas sugiere que en esta pestaña se podrán modificar comportamientos internos de la interfaz como la comunicación serial Sin embargo, dentro de ésta está previsto mostrar información referente al nivel del estanque y el estado de la batería del sistema.

Otro aspecto importante de navegación, es dar información al usuario de la pestaña que actualmente está observando. Ambas versiones se valen de modificar el aspecto del enlace que conduce a esta pestaña. Sin embargo, esta forma de representación pareciera ser insuficiente. En particular, esta situación se podría solucionar agregando el nombre de la pestaña en una de las esquinas bajo el indicador del estado de la central.

En cuanto al aspecto de la interfaz durante la segunda versión, ésta parece muy poco amigable y el uso del color rojo en algunas instancias en conjunto con la sobrecarga de información. El usuario tenía una sensación de una complejidad muy elevada. En el caso de la tercera versión, el color predominante es el blanco seguido el azul, lo que la hace más amigable. Sin embargo, la ausencia de otros colores le confiere un aspecto plano. En particular, se sugiere la utilización de otros colores para destacar algunas variables al interior de ésta y conferirle más “vida” a la interfaz.

Por otro lado, la eliminación de variables innecesarias en la interfaz mejora notablemente la usabilidad de ésta. En la versión dos, el usuario era capaz de modificar las referencias internas de los controladores de tensión y factor de potencia. Esto aumenta considerablemente la complejidad de las tareas que debe realizar el operador, aumentando la probabilidad de que este cometiera un error. Sin embargo, en el nuevo diseño, el usuario no es capaz de modificar estos campos, lo sque tampoco son crítcos para una operación normal, disminuyendo la cantidad de errores de operación posibles. Este hecho destaca la importancia de un diseño orientado a la eliminación de errores por parte del usuario.

Capítulo 5. Conclusión y Trabajo Futuro

De este trabajo es posible destacar la flexibilidad del sistema de control de la unidad. El complejo DSP-HMI provee una gama de oportunidades de desarrollo extraordinarias. Por un lado el DSP es una unidad relativamente sencilla de programar, la cual permite controlar sistemas tan complejos como lo es la unidad micro hidráulica Plug & Play a través de sus múltiples periféricos, además al incluir una HMI provee al usuario una manera agradable y sencilla de monitorear e interactuar con el sistema.

En cuanto al DSP, falta por añadir un sistema de reinicio para situaciones de fallas en el sistema. Si bien el sistema WatchDog se encuentra funcionando se deben agregar las rutinas necesarias para extraer el estado de operación de la interfaz. A su vez esto está acompañado de la implementación de la "inteligencia" de la unidad, parte de la cual se encuentra en las transiciones de falla, pero, también se debe programar la "intención" de sincronizarse o de operar en isla (operación automática normal).

Una de las acciones más importantes que debe realizar el control supervisor, es realizar los procesos de transición entre los distintos estados de operación. Si bien la mayoría de estas transiciones se encuentran implementadas casi en su totalidad, falta la señal de confirmación de cierre o apertura de contactores. Las transiciones que involucran apertura y cierre de la válvula de mariposa se encuentran incompletas. Sin embargo, dentro del código del DSP se habilitan espacios específicos para agregar los controladores de este dispositivo.

En cuanto a los estados de operación, cabe destacar que los estados utilizados son genéricos para unidades de generación distribuida, ya que cualquier unidad debe tener por lo menos los estados: Detenido, En Espera y Sincronizado. Los estados Isla, Red y Mantención, son una expansión de los estados básicos de una unidad que le confieren más funcionalidades a la unidad. Son fácilmente eliminables sin afectar la operación básica de la central. Esta cualidad permitiría intercambiar el dispositivo de generación manteniendo la estructura general del control supervisor. Obviamente la mayoría de los controles específicos quedarían inutilizados ya que están diseñados para un generador sincrónico con rotor bobinado. Por otra parte, la estructura de la HMI se podría mantener casi completamente intacta ya que hay muy pocos elementos que dependan del tipo de recuso utilizado.

Respecto a la HMI se debe destacar la importancia de paralelizar procesos gráficos y de comunicación, dado que si estos coexisten en un solo hilo de ejecución, las características gráficas de la aplicación tenderán a congelarse mientras se esté ejecutando la sección de comunicación del código. Por otra parte si el dispositivo en donde se ejecuta esta aplicación tiene características de hardware muy básicas como lo ocurrido en la SBC, esta estrategia de paralelizar funciones no operará correctamente y nuevamente aparecerán problemas como retardos o congelaciones de la interfaz.

También cabe mencionar la importancia de ponerse en situaciones totalmente inesperadas e incluso a veces ilógicas, ya que en un principio no se puede saber qué acciones tomará el usuario. Bajo el mismo marco, es importante procurar que el uso de esta sea lo más sencillo posible, de lo contrario cabe la posibilidad de confundir al usuario y por tanto aumente la probabilidad de que éste comenta errores en la operación.

Además de procurar que la interfaz tenga un aspecto sencillo se debe proveer al usuario siempre de todas las opciones de navegación de una forma clara, ya que ésta influye de manera importante en cómo éste percibe la interfaz.

Finalmente, las pruebas realizadas con la Tablet demostraron que es factible utilizar este dispositivo como HMI. Sin embargo se requerirá de un hardware especializado que alimente tanto la tablet como el adaptador serial-USB. Este hardware debe incluir un sistema de reinicio del adaptador, el cual puede ser disparado desde el DSP.

Un aspecto importante que puede generar preocupación en el usuario es la presentación de variables eléctricas. Actualmente, la mayoría de las variables se despliegan de forma relativamente estable. Sin embargo, al operar de forma sincronizada a la red, las potencias mostradas en la HMI presentan oscilaciones importantes, por lo que se siguiere mejorar el control de factor de potencia para estabilizar estas mediciones.

En cuanto a la interfaz queda por implementar la sección de alertas, ya que si bien se avanzó en los métodos de guardado de éstas, resta implementar el envío de mensajes desde el DSP y la recepción desde la HMI.

De manera general, también cabe destacar la importancia de encontrar una solución para la alimentación de los contactores del sistema. Una posible solución podría ser cambiar los contactores actuales por unos que se puedan alimentar desde una fuente DC.

Bibliografía

[1] Ministerio de Energía, «Estrategia Nacional de Energía,» 2012. [En línea]. Available: http://www.minenergia.cl/estrategia-nacional-de-energia-2012.html.

[2] Comité Técnico de la Plataforma Escenarios Energéticos 2030, «Escenarios energéticos Chile 2030,» Julio 2013. [En línea]. Available: http://escenariosenergeticos.cl/wp-content/uploads/Escenarios_Energeticos_2013.pdf.

[3] Centro de Energías, «Microhidráulica Inteligente,» Santiago de Chile, 2012.

[4] P. A. Ramírez Del Barrio, Diseño e Implementación de Sistema de Control para Micro Hidráulica Plug & Play, Santiago de Chile: www.tesis.uchile.cl, 2012.

[5] Universidad de Chile FCFM Departamento de Ingeniería Eléctrica, de *Apuntes EL42C Convercion Electromecanica de la Energía.*, 2003, pp. 235-237.

[6] W. Jara Tirapegui, Introducción a las Energías Renovables No Convencionales, Endesa Chile, 2006.

[7] Texas Instruments, «F2833x Serial Communication Interface,» http://www.ti.com/lit/ug/sprug03b/sprug03b.pdf.

[8] A. Romero Medina, «Ergonomiía cognitiva y usabilidad.,» [En línea]. Available: http://www.um.es/docencia/agustinr/Tema6-0607a.pdf.

[9] J. Márquez Correa, «Guía para evaluacion experta,» [En línea]. Available: http://www.jmarquez.com/documentos/jm_checklist.pdf.

[10] QT Nokia, «QT Project,» Nokia, [En línea]. Available: qt-project.org.

[11] «Lenovo A1000,» [En línea]. Available: https://www.pcfactory.cl/producto/14854-Tablet.IdeaTab.A1000.Cortex-A9.dual.core.1,2.GHz.1GB.4GB.7.Android.4.1.Blanca.

[12] «Lenovo S6000,» [En línea]. Available: https://www.pcfactory.cl/producto/15054-Tablet.IdeaTab.S6000.quad.core.1,2.GHz.1GB.16GB.10,1.IPS.Android.4.2.negra.

[13] «SBC TS-TPC-8390-4710,» [En línea]. Available: http://www.embeddedarm.com/products/board-detail.php?product=TS-TPC-8390-4710.

[14] Slickdev Labs, «Slick USB 2 Serial Library,» [En línea]. Available: http://slickdevlabs.com/slick-usb-2-serial-library/.

A. Programa del DSP

A.1. Programa Principal: uHidroV1-6.c

```
/*
 *
 * uhidro.c
 *
 *  Created on: 10-10-2013
 *  Update on : 14-10-2013
 *      Author: A1
 */
#include "DSP28x_Project.h"
#include "math.h"
#include "time.h"
#include "string.h"
#include "stdio.h"

//definiciones de constantes
#define UNODIVRAIZTRES  0.5773502691896258
#define DOS_PI                  6.2831853071795865
#define DOSDIVTRES              0.6666666666666666

//Contactores GEN y RED
#define KGEN_ABRIR      GpioDataRegs.GPBSET.bit.GPIO58=1;
#define KRED_ABRIR      GpioDataRegs.GPBSET.bit.GPIO60=1;
#define KGEN_CERRAR     GpioDataRegs.GPBCLEAR.bit.GPIO58=1;
#define KRED_CERRAR     GpioDataRegs.GPBCLEAR.bit.GPIO60=1;
//#define KLOAD_ABRIR   GpioDataRegs.GPBSET.bit.GPIO60=1;               // VERIFICAR gpio
//#define KLOAD_CERRAR  GpioDataRegs.GPBCLEAR.bit.GPIO60=1;

//Contrlo de frecuencia
#define CONTROL_RECT_ON GpioDataRegs.GPASET.bit.GPIO6 = 1;
#define CONTROL_RECT_OFF GpioDataRegs.GPACLEAR.bit.GPIO6 = 1;

#define CONTROL_SINC_OFF GpioDataRegs.GPASET.bit.GPIO2 = 1;             //FREC 46   ISLA1:
// AL CAMBIAR DE ESTADOS TB PUEDE MANDANDAR SEÑAL SAVE
#define CONTROL_SINC_ON GpioDataRegs.GPACLEAR.bit.GPIO2 = 1;            //FREC 50       SINC
        0:

//Flash Variables
extern Uint16 RamfuncsLoadStart;
extern Uint16 RamfuncsLoadEnd;
extern Uint16 RamfuncsRunStart;
extern void InitFlash(void);

//Macros del sistema
//PWM
#define EPWM_TIMER_TBPRD        17858
#define EPWM1_TBPRD_2K          586
#define EPWM1_TBPRD_1K          1172
#define EPWM1_TBPRD_05K 2344


// Funciones del Sistema
extern void InitSysCtrl(void);
extern void ConfigADCregister(void);
extern void ConfigGPIOregister(void);
void ConfigurarEPWM(void);
void SCIA_init();
```

```
//flujo del codigo
int adc_isr_cnt = 0,  timer_isr_cnt = 0, pwm5_isr_cnt = 0;
int tran_cnt = 0, est_cnt = 0;

//INTERRUPT FUNCTIONS
interrupt void epwm5_isr(void);
interrupt void adc_init_offset(void);
interrupt void cpu_timer0_isr(void);
interrupt void SCIA_RX_isr(void);
interrupt void SCIA_TX_isr(void);
interrupt void adc_isr(void);

//Funciones control de campo
void Changeduty(int dutycycle);

//Interrupcion ADC
void Rutina_ADC(void);
void Rutina_Fallas(void);
void Rutina_Identificar(void);
void Rutina_Estados(void);
void Rutina_Transiciones(void);
void Rutina_Control(void);
void Rutina_Actuador(void);

//###########################################################################################
//Rutina ADC
int Estados = 1, Estado_sig = 1;
char Transiciones[5] = {'-','-','-','-',0};
//Calculo de Offset ADC
int Offset_cnt = 0;
float offsetVgenAB = 0, offsetVgenBC = 0, offsetVredAB = 0, offsetVredBC = 0;
float offsetIgenA = 0, offsetIgenB = 0, offsetIredA = 0, offsetIredB = 0;
float KI_offset = 0.998, KP_offset = 0.001999;

//Calculo de variables electricas
float VgenAB = 0, VgenBC = 0,VgenCA = 0, VredAB = 0, VredBC = 0, VredCA = 0;
float VgenA = 0, VgenB = 0, VgenC = 0, VredA = 0, VredB = 0, VredC = 0;
float KgainVgenAB = 0.245252001, KgainVgenBC = 0.2457103, KgainVredAB = 0.2522306, KgainVredBC =
-0.254730444;
float VgenAB_2 = 0, VgenBC_2 = 0, VgenCA_2 = 0, VredAB_2 = 0, VredBC_2 = 0, VredCA_2 = 0;
float VgenAB_f = 0, VgenAB_22 = 0, VgenAB_21 = 0, VgenAB_f2 = 0, VgenAB_f1 = 0;
float VgenBC_f = 0, VgenBC_22 = 0, VgenBC_21 = 0, VgenBC_f2 = 0, VgenBC_f1 = 0;
float VgenCA_f = 0, VgenCA_22 = 0, VgenCA_21 = 0, VgenCA_f2 = 0, VgenCA_f1 = 0;
float VredAB_f = 0, VredAB_22 = 0, VredAB_21 = 0, VredAB_f2 = 0, VredAB_f1 = 0;
float VredBC_f = 0, VredBC_22 = 0, VredBC_21 = 0, VredBC_f2 = 0, VredBC_f1 = 0;
float VredCA_f = 0, VredCA_22 = 0, VredCA_21 = 0, VredCA_f2 = 0, VredCA_f1 = 0;
float VgenAB_rms = 0, VgenBC_rms = 0, VgenCA_rms = 0, VredAB_rms = 0, VredBC_rms = 0, VredCA_rms
= 0;
float VgenA_rms = 0, VgenB_rms = 0, VgenC_rms = 0, VredA_rms = 0, VredB_rms = 0, VredC_rms = 0;
float Vgenrms = 0, Vredrms = 0;

float IgenA = 0, IgenB = 0, IgenC = 0, IredA = 0, IredB = 0, IredC = 0;
float KgainIgenA = -0.01233981, KgainIgenB = -0.0123, KgainIredA = -0.01220081, KgainIredB = -
0.0121826;
float IgenA_2 = 0, IgenB_2 = 0, IgenC_2 = 0, IredA_2 = 0, IredB_2 = 0, IredC_2 = 0;
float IgenA_f = 0, IgenA_22 = 0, IgenA_21 = 0, IgenA_f2 = 0, IgenA_f1 = 0;
float IgenB_f = 0, IgenB_22 = 0, IgenB_21 = 0, IgenB_f2 = 0, IgenB_f1 = 0;
float IgenC_f = 0, IgenC_22 = 0, IgenC_21 = 0, IgenC_f2 = 0, IgenC_f1 = 0;
float IredA_f = 0, IredA_22 = 0, IredA_21 = 0, IredA_f2 = 0, IredA_f1 = 0;
float IredB_f = 0, IredB_22 = 0, IredB_21 = 0, IredB_f2 = 0, IredB_f1 = 0;
float IredC_f = 0, IredC_22 = 0, IredC_21 = 0, IredC_f2 = 0, IredC_f1 = 0;
float IgenA_rms = 0, IgenB_rms = 0, IgenC_rms = 0, IredA_rms = 0, IredB_rms = 0, IredC_rms = 0;
float Igenrms = 0, Iredrms = 0;

//Transformada Alfa-Beta
float Vgenalfa = 0, Vgenbeta = 0, Vredalfa = 0, Vredbeta = 0;
float Igenalfa = 0, Igenbeta = 0, Iredalfa = 0, Iredbeta = 0;

//PLL
```

```
float VMgen = 0, VMred = 0;
float Chigen = 0, Chired = 0;
float Theta_PLLgen = 0, Theta_PLLred = 0, Theta_PLL0gen = 0, Theta_PLL0red = 0;
float oPLL1gen = 0, oPLL0gen = 0, we_PLLgen = 0;
float oPLL1red = 0, oPLL0red = 0, we_PLLred = 0;
float Fe_PLLgen = 0, Fe_PLLgen1 = 0, Fe_PLLgen2 = 0, Fe_PLLred = 0, Fe_PLLred1 = 0, Fe_PLLred2 =
0;
float Fe_PLLgen_f = 0, Fe_PLLgen_f1 = 0, Fe_PLLgen_f2 = 0, Fe_PLLred_f = 0, Fe_PLLred_f1 = 0,
Fe_PLLred_f2 = 0;

//dq
float Vdgen = 0, Vqgen = 0, Vdred = 0, Vqred = 0;
float Idgen = 0, Iqgen = 0, Idred = 0, Iqred = 0;
float Vdgen_f = 0, Vdgen_f1 = 0, Vdgen_f2 = 0, Vdgen1 = 0, Vdgen2 = 0;
float Vqgen_f = 0, Vqgen_f1 = 0, Vqgen_f2 = 0, Vqgen1 = 0, Vqgen2 = 0;
float Idgen_f = 0, Idgen_f1 = 0, Idgen_f2 = 0, Idgen1 = 0, Idgen2 = 0;
float Iqgen_f = 0, Iqgen_f1 = 0, Iqgen_f2 = 0, Iqgen1 = 0, Iqgen2 = 0;

//Potencia
float Pgen = 0, Qgen = 0, Pred = 0, Qred = 0;
float FPgen = 0, FPred = 0, Sgen = 0, Sred =0;
float Pgen_f = 0, Pgen_f1 = 0, Pgen_f2 = 0, Pgen1 = 0, Pgen2 = 0;
float Qgen_f = 0, Qgen_f1 = 0, Qgen_f2 = 0, Qgen1 = 0, Qgen2 = 0;
float Pred_f = 0, Pred_f1 = 0, Pred_f2 = 0, Pred1 = 0, Pred2 = 0;
float Qred_f = 0, Qred_f1 = 0, Qred_f2 = 0, Qred1 = 0, Qred2 = 0;
int Phi = -1;
float Phir = 0;

//#############################################################################################
//Rutina Fallas
int flag = 0, flag_clc = 0;
int Cancel_all = 0, Cancel_tn = 0;


//#############################################################################################
//Rutina Estados
int VF_estado = 0;              // 0: OFF, 1: tension, 2: factor de potencia
float VF_ref = 0, PF_ref = 0.99, Phir_ref = -0.451;
float VF_ch = 220, PF_ch = 0.9, Phir_ch = -0.451;
int  Phi_ref = -1, Phi_ch = -1;
int EValvula_ch = 100;

//#############################################################################################
//Rutina Transiciones
int VFcnt = 0, Theta_cnt = 0, Cancel = 0;
//#############################################################################################
//Rutina Control
int Duty_max = 0, Duty_min = 0, Duty = 0;
float errVgen = 0, IntVgen0 = 0, IntVgen1 = 0, KIVgen = 0.025, KPVgen = 50, Vf = 0, Vf_max =
5500, Vf_min = 300, Vf_max2 = 5500, Vf_min2 = 3100, KDuty = 4.59;
float errPFgen = 0, IntPFgen0 = 4000, IntPFgen1 = 0, KIPFgen = 1, KPPFgen = 0.0002, Igenref = 1,
Igen_max = 15, Igen_min = 0.1; // KPPFgen = 1.56 KIPF = 0.0057
float errIgen = 0, IntIgen0 = 0, IntIgen1 = 0, KIIgen = 0.25, KPIgen = 500;
float VfI = 0, Igenref2 = 0;
int Pgen_fcnt = 0;
float ProPFgen1 = 0;
int PF_cnt = 50;
float DeltaVf = 5;
//#############################################################################################
//Rutina Actuador
//Estados de contactores y Valvula
// 0 es abierto, 1 es cerrado
int  EKgen = 0, EKred = 0, EKload = 0, EValvula = 0, EFrec = 0, EFsinc = 0;

//#############################################################################################
//Comunicacion SCI
char msg[15];
char Brx[16];
void leer_rx(void);
```

```
int Estado_TX = 1, id_TX=1, Estado_falla = 0;
void build_msgS(void);  // rms general
void build_msgG(void);          // Generador
void build_msgR(void);          // Red
void build_msgFA(void); // Falla

//#############################################################################################
// Variables desechables
float Duty_max_ref = 0.59;
float Eg = 0, Xad = 264, Ifm = 0, Vfm = 0, Rfm = 7.20;
int Ten = 0;

//#############################################################################################
//      Main
void main(void){

        // Device init
        InitSysCtrl();
        EALLOW;
        SysCtrlRegs.HISPCP.all = 0x2;  //37.5 MHz
        EDIS;

        //Flash
        memcpy(&RamfuncsRunStart, &RamfuncsLoadStart, &RamfuncsLoadEnd - &RamfuncsLoadStart);
        InitFlash();

        //GPIO
        EALLOW;
        ConfigGPIOregister();
        EDIS;

        //Abrir contactores
        KGEN_ABRIR;
        KRED_ABRIR;

        //Interrupt
        DINT;
        InitPieCtrl();
        IER = 0x0000;
        IFR = 0x0000;
        InitPieVectTable();

        //Timer
        InitCpuTimers();
        ConfigCpuTimer(&CpuTimer0, 150, 300000);

        //PWM
        ConfigurarEPWM();

        //ADC
        InitAdc();
        ConfigADCregister();

        //Comunicacion
        SCIA_init();

        //Interrupt

        EALLOW;
        PieVectTable.EPWM5_INT = &epwm5_isr;
        PieVectTable.ADCINT = &adc_init_offset;
        PieVectTable.TINT0 = &cpu_timer0_isr;
        PieVectTable.SCIRXINTA = &SCIA_RX_isr;
        PieVectTable.SCITXINTA = &SCIA_TX_isr;
        EDIS;

        // Habilitar interrupciones
        PieCtrlRegs.PIEIER1.bit.INTx6 = 1; //ADC
        PieCtrlRegs.PIEIER1.bit.INTx7 = 1; //TIMER
```

```
        IER |= M_INT1;
        PieCtrlRegs.PIEIER3.bit.INTx5 = 1; //PWM
        IER |= M_INT3;
        PieCtrlRegs.PIEIER9.bit.INTx1 = 1; // RX_A
        PieCtrlRegs.PIEIER9.bit.INTx2 = 1; // TX_A
        IER |= M_INT9;

        EINT;
        ERTM;

        //Chopper del campo
        Changeduty(400);

        //timer go
        CpuTimer0Regs.TCR.bit.TSS = 0;

    for(;;)
    {
           asm("          NOP");
    }

}


void ConfigurarEPWM(void){
        EALLOW;
        SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0; // Deshabilita el reloj del EPWM
        EDIS;

        //-------------------------------------------------------------------------------------------
-----------------------------------
        //Configuración Periodo PWM
        //-------------------------------------------------------------------------------------------
-----------------------------------
        //
        //    TBPRD=            TPWM
        //              ---------------------------------------
        //                                2* TSYS(150MHZ)*2^(HSPCLKDIV)*2^(CLKDIV)
        //
        // En el caso de 5 y 6 TBPRD=17858  y  TPWM=1/FrecPWM  ---------
>FrecPWM=4199,798409676335535894277074700 4
        // En el caso de 1,2 y 3 TBPRD=2000 -------.>FrecPWM=586  <----------revisar si funca
mejor con frec 1176
        //
        //-------------------------------------------------------------------------------------------
-----------------------------------
        // >>>>> CONFIGURACIÓN EPWM5 <<<<<
        //-------------------------------------------------------------------------------------------
-----------------------------------
        // Setup TBCLK
        EPwm5Regs.TBPRD = EPWM_TIMER_TBPRD;           // Fijar período del PWM
        EPwm5Regs.TBCTR = 0x0000;                      // Clear counter
        // Setup counter mode
        EPwm5Regs.TBCTL.bit.CTRMODE = 2; // Modo simétrico (ie contador triangular)
        EPwm5Regs.TBCTL.bit.HSPCLKDIV = 0;       // TBLK = SYSCLKOUT/(CLKDIV x HSPCLKDIV) este
registro es pa tener un control más fino de la frecuencia
        EPwm5Regs.TBCTL.bit.CLKDIV = 0;          // Clock ratio to SYSCLKOUT en este caso:
TB_DIV1 -> TBLK = SYSCLKOUT
        // Set actions
        EPwm5Regs.AQCTLB.all                                    = 0x0600;
        // Interrupt where we will change the Compare Values
        EPwm5Regs.ETSEL.bit.INTSEL = ET_CTR_ZERO;      // Select INT on Zero event
        EPwm5Regs.ETSEL.bit.INTEN = 1;                 // Enable INT
        EPwm5Regs.ETPS.bit.INTPRD = ET_1ST;            // Genera interrupción cada vez

        //-------------------------------------------------------------------------------------------
-----------------------------------
        // >>>>> CONFIGURACIÓN EPWM 1<<<<<
```

```
        //-----------------------------------------------------------------------------------------
-------------------------------------
        // Setup TBCLK
        EPwm1Regs.TBCTL.bit.CLKDIV                              = 3;
        EPwm1Regs.TBCTL.bit.HSPCLKDIV                   = 4;
        EPwm1Regs.TBCTL.bit.CTRMODE                     = 2;
        EPwm1Regs.AQCTLB.all                                    = 0x0600;
        EPwm1Regs.TBPRD                                         = EPWM1_TBPRD_05K;
        // aprox 1KHz - PWM signal
        Duty_max = (int) (EPWM1_TBPRD_05K*0.59);
        Duty_min = (int) (EPWM1_TBPRD_05K*0.03);
        //-----------------------------------------------------------------------------------------
-------------------------------------
        //Comienza con CMPB (ciclo de trabajo) en cero con ciclo máximo y puede aumentar hasta
el maximo de 2000 obteniendo un cero
        //Se debe considerar que la logica es inversa debido a la electrónica con que se
implementó el prototipo
        //-----------------------------------------------------------------------------------------
-------------------------------------

        EPwm1Regs.CMPB                                          = 0;

        EALLOW;
        SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;  // Habilitar el reloj del EPWM
        EDIS;

}

void SCIA_init()
{
        SciaRegs.SCICCR.all =0x0007;    // 1 stop bit,  No loopback
                                        // ODD parity,8 char bits,
                                        // async mode, idle-line protocol
        SciaRegs.SCICTL1.all =0x0003;   // enable TX, RX, internal SCICLK,
                                        // Disable RX ERR, SLEEP, TXWAKE

        // SYSCLOCKOUT = 150MHz; LSPCLK = 1/4 = 37.5 MHz
        // BRR = (LSPCLK / (9600 x 8)) -1
        // BRR = 487  gives 9605 Baud
        SciaRegs.SCIHBAUD    = 487 >> 8;        // Highbyte
        SciaRegs.SCILBAUD    = 487 & 0x00FF;    // Lowbyte

        SciaRegs.SCICTL2.bit.TXINTENA = 1;              // enable SCI-A Tx-ISR
        SciaRegs.SCICTL2.bit.RXBKINTENA = 1;    // enable SCI_A Rx-ISR

        SciaRegs.SCIFFTX.all = 0xC060;  // bit 15 = 1 : relinquish from Reset
                                                                                // bit 14 = 1 : Enable
FIFO
                                                                                // bit 6 = 1 :  CLR
TXFFINT-Flag
                                                                                // bit 5 = 1 :  enable
TX FIFO match
                                                                                // bit 4-0 :  TX-ISR, if
TX FIFO is 0(empty)
        SciaRegs.SCIFFCT.all = 0x0000;  // Set FIFO transfer delay to 0

        SciaRegs.SCIFFRX.all = 0xE065;  // Rx interrupt level = 5

        SciaRegs.SCICTL1.all = 0x0023;  // Relinquish SCI from Reset
}

interrupt void epwm5_isr(void) // Inicio de ciclo. Aquí se hace SOC y se fija la bandera que usa
la interrupción del adc pa saber en q parte del pwm estamos
{
        pwm5_isr_cnt++;
        AdcRegs.ADCTRL2.bit.SOC_SEQ1 = 1; // Comanda el inicio de la adquisición de datos
        GpioDataRegs.GPATOGGLE.bit.GPIO15 = 1;
        //GpioDataRegs.GPASET.bit.PIN_PROCESAMIENTO = 1; // Para ver cuanto tiempo de
procesamiento está usado
```

```
        // Clear INT flag for this timer
        EPwm5Regs.ETCLR.bit.INT = 1;

        // Acknowledge this interrupt to receive more interrupts from group 3
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP3;

}

interrupt void adc_init_offset(void){

        Offset_cnt++;

        offsetVgenAB = KI_offset*offsetVgenAB+KP_offset*((float)(AdcRegs.ADCRESULT4 >>4));
        offsetVgenBC = KI_offset*offsetVgenBC+KP_offset*((float)(AdcRegs.ADCRESULT8 >>4));
        offsetVredAB = KI_offset*offsetVredAB+KP_offset*((float)(AdcRegs.ADCRESULT0 >>4));
        offsetVredBC = KI_offset*offsetVredBC+KP_offset*((float)(AdcRegs.ADCRESULT2 >>4));

        offsetIgenA = KI_offset*offsetIgenA+KP_offset*((float)(AdcRegs.ADCRESULT6 >>4));
        offsetIgenB = KI_offset*offsetIgenB+KP_offset*((float)(AdcRegs.ADCRESULT7 >>4));
        offsetIredA = KI_offset*offsetIredA+KP_offset*((float)(AdcRegs.ADCRESULT3 >>4));
        offsetIredB = KI_offset*offsetIredB+KP_offset*((float)(AdcRegs.ADCRESULT1 >>4));

        if(Offset_cnt >= 30000){
                EALLOW;
                PieVectTable.ADCINT = &adc_isr;
                EDIS;
        }

        //Limpiar ADC
        AdcRegs.ADCTRL2.bit.RST_SEQ1 = 0x1;     // Resetea el contador de SEQ1
        AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;             // Clear INT SEQ1 bit
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; // Acknowledge interrupt to PIE
}

interrupt void adc_isr(void){
        adc_isr_cnt++;

        Rutina_ADC();
        Rutina_Fallas();

        if(Estados != Estado_sig && strncmp(Transiciones,"----",4) == 0)
                Rutina_Identificar();

        if(Estados > 0 && strncmp(Transiciones,"----",4) == 0){
                Rutina_Estados();
                est_cnt++;
        }
        else if(Estados != Estado_sig && strncmp(Transiciones,"----",4) != 0){
                Rutina_Transiciones();
                tran_cnt++;
        }
        Rutina_Control();
        Rutina_Actuador();

        //Limpiar ADC
        AdcRegs.ADCTRL2.bit.RST_SEQ1 = 0x1;     // Resetea el contador de SEQ1
        AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;             // Clear INT SEQ1 bit
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; // Acknowledge interrupt to PIE
}

void Rutina_ADC(void){
                //Lectura ADC
                VgenAB=KgainVgenAB*((float)(AdcRegs.ADCRESULT4 >>4)-offsetVgenAB);
                VgenBC=KgainVgenBC*((float)(AdcRegs.ADCRESULT8 >>4)-offsetVgenBC);
                VredAB=KgainVredAB*((float)(AdcRegs.ADCRESULT0 >>4)-offsetVredAB);
                VredBC=KgainVredBC*((float)(AdcRegs.ADCRESULT2 >>4)-offsetVredBC);

                IgenA=KgainIgenA*((float)(AdcRegs.ADCRESULT6 >>4)-offsetIgenA);
```

```
                IgenB=KgainIgenB*((float)(AdcRegs.ADCRESULT7 >>4)-offsetIgenB);
                IredA=KgainIredA*((float)(AdcRegs.ADCRESULT3 >>4)-offsetIredA);
                IredB=KgainIredB*((float)(AdcRegs.ADCRESULT1 >>4)-offsetIredB);

                //Fase C
                VgenCA=0-(VgenAB+VgenBC);
                VredCA=0-(VredAB+VredBC);
                IgenC = - IgenA - IgenB;
                IredC = - IredA - IredB;

                //Fase-Fase a Fase-Neutro
                VgenA=0.333333333333333*(VgenAB-VgenCA);
                VgenB=0.333333333333333*(VgenBC-VgenAB);
                VgenC=0.333333333333333*(VgenCA-VgenBC);
                VredA=0.333333333333333*(VredAB-VredCA);
                VredB=0.333333333333333*(VredBC-VredAB);
                VredC=0.333333333333333*(VredCA-VredBC);

                //RMS
                VgenAB_2 = VgenAB*VgenAB;
                VgenBC_2 = VgenBC*VgenBC;
                VgenCA_2 = VgenCA*VgenCA;
                VredAB_2 = VredAB*VredAB;
                VredBC_2 = VredBC*VredBC;
                VredCA_2 = VredCA*VredCA;

                VgenAB_f=8.85174031474353e-005*VgenAB_21+8.85174031474353e-005*VgenAB_22-(-
1.98109467563163*VgenAB_f1+0.98127171043792*VgenAB_f2);
                VgenAB_22=VgenAB_21;
                VgenAB_21=VgenAB_2;
                VgenAB_f2=VgenAB_f1;
                VgenAB_f1=VgenAB_f;

                VgenBC_f=8.85174031474353e-005*VgenBC_21+8.85174031474353e-005*VgenBC_22-(-
1.98109467563163*VgenBC_f1+0.98127171043792*VgenBC_f2);
                VgenBC_22=VgenBC_21;
                VgenBC_21=VgenBC_2;
                VgenBC_f2=VgenBC_f1;
                VgenBC_f1=VgenBC_f;

                VgenCA_f=8.85174031474353e-005*VgenCA_21+8.85174031474353e-005*VgenCA_22-(-
1.98109467563163*VgenCA_f1+0.98127171043792*VgenCA_f2);
                VgenCA_22=VgenCA_21;
                VgenCA_21=VgenCA_2;
                VgenCA_f2=VgenCA_f1;
                VgenCA_f1=VgenCA_f;

                VredAB_f=8.85174031474353e-005*VredAB_21+8.85174031474353e-005*VredAB_22-(-
1.98109467563163*VredAB_f1+0.98127171043792*VredAB_f2);
                VredAB_22=VredAB_21;
                VredAB_21=VredAB_2;
                VredAB_f2=VredAB_f1;
                VredAB_f1=VredAB_f;

                VredBC_f=8.85174031474353e-005*VredBC_21+8.85174031474353e-005*VredBC_22-(-
1.98109467563163*VredBC_f1+0.98127171043792*VredBC_f2);
                VredBC_22=VredBC_21;
                VredBC_21=VredBC_2;
                VredBC_f2=VredBC_f1;
                VredBC_f1=VredBC_f;

                VredCA_f=8.85174031474353e-005*VredCA_21+8.85174031474353e-005*VredCA_22-(-
1.98109467563163*VredCA_f1+0.98127171043792*VredCA_f2);
                VredCA_22=VredCA_21;
                VredCA_21=VredCA_2;
                VredCA_f2=VredCA_f1;
                VredCA_f1=VredCA_f;

                VgenAB_rms = sqrt(VgenAB_f);
```

```
                VgenBC_rms = sqrt(VgenBC_f);
                VgenCA_rms = sqrt(VgenCA_f);
                VredAB_rms = sqrt(VredAB_f);
                VredBC_rms = sqrt(VredBC_f);
                VredCA_rms = sqrt(VredCA_f);

                VgenA_rms = UNODIVRAIZTRES*(VgenAB_rms+VgenCA_rms)*0.5;
                VgenB_rms = UNODIVRAIZTRES*(VgenBC_rms+VgenAB_rms)*0.5;
                VgenC_rms = UNODIVRAIZTRES*(VgenCA_rms+VgenBC_rms)*0.5;
                VredA_rms = UNODIVRAIZTRES*(VredAB_rms+VredCA_rms)*0.5;
                VredB_rms = UNODIVRAIZTRES*(VredBC_rms+VredAB_rms)*0.5;
                VredC_rms = UNODIVRAIZTRES*(VredCA_rms+VredBC_rms)*0.5;

                Vgenrms = (VgenAB_rms + VgenBC_rms + VgenCA_rms)*UNODIVRAIZTRES/3;
                Vredrms = (VredAB_rms + VredBC_rms + VredCA_rms)*UNODIVRAIZTRES/3;

                IgenA_2 = IgenA*IgenA;
                IgenB_2 = IgenB*IgenB;
                IgenC_2 = IgenC*IgenC;
                IredA_2 = IredA*IredA;
                IredB_2 = IredB*IredB;
                IredC_2 = IredC*IredC;

                IgenA_f=8.85174031474353e-005*IgenA_21+8.85174031474353e-005*IgenA_22-(-
1.98109467563163*IgenA_f1+0.98127171043792*IgenA_f2);
                IgenA_22=IgenA_21;
                IgenA_21=IgenA_2;
                IgenA_f2=IgenA_f1;
                IgenA_f1=IgenA_f;

                IgenB_f=8.85174031474353e-005*IgenB_21+8.85174031474353e-005*IgenB_22-(-
1.98109467563163*IgenB_f1+0.98127171043792*IgenB_f2);
                IgenB_22=IgenB_21;
                IgenB_21=IgenB_2;
                IgenB_f2=IgenB_f1;
                IgenB_f1=IgenB_f;

                IgenC_f=8.85174031474353e-005*IgenC_21+8.85174031474353e-005*IgenC_22-(-
1.98109467563163*IgenC_f1+0.98127171043792*IgenC_f2);
                IgenC_22=IgenC_21;
                IgenC_21=IgenC_2;
                IgenC_f2=IgenC_f1;
                IgenC_f1=IgenC_f;

                IredA_f=8.85174031474353e-005*IredA_21+8.85174031474353e-005*IredA_22-(-
1.98109467563163*IredA_f1+0.98127171043792*IredA_f2);
                IredA_22=IredA_21;
                IredA_21=IredA_2;
                IredA_f2=IredA_f1;
                IredA_f1=IredA_f;

                IredB_f=8.85174031474353e-005*IredB_21+8.85174031474353e-005*IredB_22-(-
1.98109467563163*IredB_f1+0.98127171043792*IredB_f2);
                IredB_22=IredB_21;
                IredB_21=IredB_2;
                IredB_f2=IredB_f1;
                IredB_f1=IredB_f;

                IredC_f=8.85174031474353e-005*IredC_21+8.85174031474353e-005*IredC_22-(-
1.98109467563163*IredC_f1+0.98127171043792*IredC_f2);
                IredC_22=IredC_21;
                IredC_21=IredC_2;
                IredC_f2=IredC_f1;
                IredC_f1=IredC_f;

                IgenA_rms = sqrt(IgenA_f);
                IgenB_rms = sqrt(IgenB_f);
                IgenC_rms = sqrt(IgenC_f);
                IredA_rms = sqrt(IredA_f);
```

```
                IredB_rms = sqrt(IredB_f);
                IredC_rms = sqrt(IredC_f);

                Igenrms = (IgenA_rms + IgenB_rms + IgenC_rms)/3;
                Iredrms = (IredA_rms + IredB_rms + IredC_rms)/3;

                //Transformada Alfa-Beta
                Vgenalfa=VgenA - 0.5*(VgenB+VgenC);
                Vgenbeta=0.866025403784439*(VgenB-VgenC);
                Vredalfa=VredA - 0.5*(VredB+VredC);
                Vredbeta=0.866025403784439*(VredB-VredC);

                Igenalfa=IgenA - 0.5*(IgenB+IgenC);
                Igenbeta=0.866025403784439*(IgenB-IgenC);
                Iredalfa=IredA - 0.5*(IredB+IredC);
                Iredbeta=0.866025403784439*(IredB-IredC);

                //PLL
                VMgen = sqrt(Vgenalfa*Vgenalfa + Vgenbeta*Vgenbeta);
                VMred = sqrt(Vredalfa*Vredalfa + Vredbeta*Vredbeta);

                if (VMgen<1) VMgen = 1;
                if (VMred<1) VMred = 1;

                Chigen = (Vgenbeta*cos(Theta_PLLgen) - Vgenalfa*sin(Theta_PLLgen))/VMgen;
                Chired = (Vredbeta*cos(Theta_PLLred) - Vredalfa*sin(Theta_PLLred))/VMred;

                oPLL1gen=oPLL0gen+ 0.846785235027261*Chigen;//0.001
                we_PLLgen=oPLL1gen+87.9287918294165*Chigen;//0.1
                oPLL0gen=oPLL1gen;
                oPLL1red=oPLL0red+ 0.846785235027261*Chired;//0.001
                we_PLLred=oPLL1red+87.9287918294165*Chired;//0.1
                oPLL0red=oPLL1red;


        Theta_PLLgen=(0.000212765957446808510638297872340 43*we_PLLgen+Theta_PLL0gen);//integrado
r

        Theta_PLLred=(0.000212765957446808510638297872340 43*we_PLLred+Theta_PLL0red);//integrado
r

                if (Theta_PLLgen>DOS_PI) Theta_PLLgen=Theta_PLLgen-DOS_PI;
                else if (Theta_PLLgen<0) Theta_PLLgen=Theta_PLLgen+DOS_PI;
                Theta_PLL0gen=Theta_PLLgen;
                if (Theta_PLLred>DOS_PI) Theta_PLLred=Theta_PLLred-DOS_PI;
                else if (Theta_PLLred<0) Theta_PLLred=Theta_PLLred+DOS_PI;
                Theta_PLL0red=Theta_PLLred;

                Fe_PLLgen=0.892857142857143*(we_PLLgen/DOS_PI); //[Hz]
                Fe_PLLgen_f=8.85174031474353e-005*Fe_PLLgen1+8.85174031474353e-005*Fe_PLLgen2-(-
1.98109467563163*Fe_PLLgen_f1+0.98127171043792*Fe_PLLgen_f2);
                Fe_PLLgen2=Fe_PLLgen1;
                Fe_PLLgen1=Fe_PLLgen;
                Fe_PLLgen_f2=Fe_PLLgen_f1;
                Fe_PLLgen_f1=Fe_PLLgen_f;
                Fe_PLLred=0.892857142857143*(we_PLLred/DOS_PI); //[Hz]
                Fe_PLLred_f=8.85174031474353e-005*Fe_PLLred1+8.85174031474353e-005*Fe_PLLred2-(-
1.98109467563163*Fe_PLLred_f1+0.98127171043792*Fe_PLLred_f2);
                Fe_PLLred2=Fe_PLLred1;
                Fe_PLLred1=Fe_PLLred;
                Fe_PLLred_f2=Fe_PLLred_f1;
                Fe_PLLred_f1=Fe_PLLred_f;

                //dq
                Vdgen = Vgenalfa*cos(Theta_PLLgen) + Vgenbeta*sin(Theta_PLLgen);
                Vqgen = - Vgenalfa*sin(Theta_PLLgen) +Vgenbeta*cos(Theta_PLLgen);
                Vdred = Vredalfa*cos(Theta_PLLred) + Vredbeta*sin(Theta_PLLred);
                Vqred = - Vredalfa*sin(Theta_PLLred) +Vredbeta*cos(Theta_PLLred);
```

```
                Idgen = Igenalfa*cos(Theta_PLLgen) + Igenbeta*sin(Theta_PLLgen);
                Iqgen = - Igenalfa*sin(Theta_PLLgen) +Igenbeta*cos(Theta_PLLgen);
                Idred = Iredalfa*cos(Theta_PLLred) + Iredbeta*sin(Theta_PLLred);
                Iqred = - Iredalfa*sin(Theta_PLLred) +Iredbeta*cos(Theta_PLLred);

                //Cambio de tranductores de tension para estado Sincronizado
                if(Estados == 3){
                        Vdgen = Vdred;
                        Vqgen = Vqred;
                }

                Vdgen_f=8.85174031474353e-005*Vdgen1+8.85174031474353e-005*Vdgen2-(-
1.98109467563163*Vdgen_f1+0.98127171043792*Vdgen_f2);
                Vdgen2=Vdgen1;
                Vdgen1=Vdgen;
                Vdgen_f2=Vdgen_f1;
                Vdgen_f1=Vdgen_f;
                Vqgen_f=8.85174031474353e-005*Vqgen1+8.85174031474353e-005*Vqgen2-(-
1.98109467563163*Vqgen_f1+0.98127171043792*Vqgen_f2);
                Vqgen2=Vqgen1;
                Vqgen1=Vqgen;
                Vqgen_f2=Vqgen_f1;
                Vqgen_f1=Vqgen_f;
                Idgen_f=8.85174031474353e-005*Idgen1+8.85174031474353e-005*Idgen2-(-
1.98109467563163*Idgen_f1+0.98127171043792*Idgen_f2);
                Idgen2=Idgen1;
                Idgen1=Idgen;
                Idgen_f2=Idgen_f1;
                Idgen_f1=Idgen_f;
                Iqgen_f=8.85174031474353e-005*Iqgen1+8.85174031474353e-005*Iqgen2-(-
1.98109467563163*Iqgen_f1+0.98127171043792*Iqgen_f2);
                Iqgen2=Iqgen1;
                Iqgen1=Iqgen;
                Iqgen_f2=Iqgen_f1;
                Iqgen_f1=Iqgen_f;

                //Potencia
                Pgen = DOSDIVTRES*(Vdgen_f*Idgen_f + Vqgen_f*Iqgen_f);
                Qgen = DOSDIVTRES*(-Vdgen_f*Iqgen_f + Vqgen_f*Idgen_f);
                Pred = DOSDIVTRES*(Vdred*Idred + Vqred*Iqred);
                Qred = DOSDIVTRES*(-Vdred*Iqred + Vqred*Idred);

                Sgen = sqrt(Pgen*Pgen+Qgen*Qgen);
                if(Sgen<1) Sgen = 1;
                Sred = sqrt(Pred*Pred+Qred*Qred);
                if(Sred<1) Sred = 1;
/*
                Pgen_f=8.85174031474353e-005*Pgen1+8.85174031474353e-005*Pgen2-(-
1.98109467563163*Pgen_f1+0.98127171043792*Pgen_f2);
                Pgen2=Pgen1;
                Pgen1=Pgen;
                Pgen_f2=Pgen_f1;
                Pgen_f1=Pgen_f;
                Qgen_f=8.85174031474353e-005*Qgen1+8.85174031474353e-005*Qgen2-(-
1.98109467563163*Qgen_f1+0.98127171043792*Qgen_f2);
                Qgen2=Qgen1;
                Qgen1=Qgen;
                Qgen_f2=Qgen_f1;
                Qgen_f1=Qgen_f;
                Pred_f=8.85174031474353e-005*Pred1+8.85174031474353e-005*Pred2-(-
1.98109467563163*Pred_f1+0.98127171043792*Pred_f2);
                Pred2=Pred1;
                Pred1=Pred;
                Pred_f2=Pred_f1;
                Pred_f1=Pred_f;
                Qred_f=8.85174031474353e-005*Qred1+8.85174031474353e-005*Qred2-(-
1.98109467563163*Qred_f1+0.98127171043792*Qred_f2);
                Qred2=Qred1;
                Qred1=Qred;
```

```
                Qred_f2=Qred_f1;
                Qred_f1=Qred_f;

                Sgen = sqrt(Pgen_f*Pgen_f+Qgen_f*Qgen_f);
                if(Sgen<1) Sgen = 1;
                Sred = sqrt(Pred_f*Pred_f+Qred_f*Qred_f);
                if(Sred<1) Sred = 1;
*/

                FPgen = Pgen/Sgen;
                FPred = Pred/Sred;

                Phir = acos(FPgen);

                Phi = 1;                                    //capacitivo
                if(Qgen > 0) Phi = -1;  //inductivo

                Phir = Phi*Phir;

                //Eg
                Vfm = 24.0*Duty/2344.0;
                Ifm = Vfm/Rfm;
                Eg  = Xad*Ifm;

}


void Rutina_Fallas(void){

        //Protecion de Motorizacion
        if(Pgen_f < -100 && Estados == 3)       Pgen_fcnt++;
        else    Pgen_fcnt=0;

        if(Estados == 3 && Pgen_fcnt >= 470){
                Estado_sig = 4;
                Pgen_fcnt = 0;
                flag |= 0x0001 ;
        }

        // Protecion de sobretension
        if(abs(VgenAB) > 600 || abs(VgenBC) > 600){
                Estado_sig = 2;
                flag |= 0x0002;
        }

        if(Cancel_all == 1){
                Estados = 1;
                Estado_sig = 1;
                Transiciones[0] = '-';
                Transiciones[1] = '-';
                Transiciones[2] = '-';
                Transiciones[3] = '-';
                Cancel_all = 0;
                flag |= 0x8000;
        }

        if(Cancel_tn == 1){
                Estado_sig = Estados;
                Transiciones[0] = '-';
                Transiciones[1] = '-';
                Transiciones[2] = '-';
                Transiciones[3] = '-';
                Cancel_tn = 0;
                flag |= 0x4000;
        }

        //flag clear
        if(flag_clc == 1){
                flag = 0;
```

```
                flag_clc = 0;
        }


}

void Rutina_Identificar(void){
        Transiciones[0] = 'T';
        Transiciones[1] = 'N';
        if(Estados == 1 && Estado_sig == 2){
                Transiciones[2] = '1';
                Transiciones[3] = '2';
        }
        else if(Estados == 1 && Estado_sig == 5){
                Transiciones[2] = '1';
                Transiciones[3] = '5';
        }
        else if(Estados == 1 && Estado_sig == 6){
                        Transiciones[2] = '1';
                        Transiciones[3] = '6';
        }
        else if(Estados == 2 && Estado_sig == 1){
                        Transiciones[2] = '2';
                        Transiciones[3] = '1';
        }
        else if(Estados == 2 && Estado_sig == 3){
                        Transiciones[2] = '2';
                        Transiciones[3] = '3';
        }
        else if(Estados == 2 && Estado_sig == 4){
                        Transiciones[2] = '2';
                        Transiciones[3] = '4';
        }
        else if(Estados == 2 && Estado_sig == 5){
                        Transiciones[2] = '2';
                        Transiciones[3] = '5';
        }
        else if(Estados == 2 && Estado_sig == 6){
                        Transiciones[2] = '2';
                        Transiciones[3] = '6';
        }
        else if(Estados == 4 && Estado_sig == 1){
                        Transiciones[2] = '4';
                        Transiciones[3] = '1';
        }
        else if(Estados == 4 && Estado_sig == 2){
                        Transiciones[2] = '4';
                        Transiciones[3] = '2';
        }
        else if(Estados == 4 && Estado_sig == 3){
                        Transiciones[2] = '4';
                        Transiciones[3] = '3';
        }
        else if(Estados == 3 && Estado_sig == 1){
                        Transiciones[2] = '3';
                        Transiciones[3] = '1';
        }
        else if(Estados == 3 && Estado_sig == 2){
                        Transiciones[2] = '3';
                        Transiciones[3] = '2';
        }
        else if(Estados == 3 && Estado_sig == 4){
                        Transiciones[2] = '3';
                        Transiciones[3] = '4';
        }
        else if(Estados == 3 && Estado_sig == 5){
                        Transiciones[2] = '3';
                        Transiciones[3] = '5';
        }
```

```
        else if(Estados == 3 && Estado_sig == 6){
                        Transiciones[2] = '3';
                        Transiciones[3] = '6';
        }
        else if(Estados == 5 && Estado_sig == 1){
                        Transiciones[2] = '5';
                        Transiciones[3] = '1';
        }
        else if(Estados == 5 && Estado_sig == 6){
                        Transiciones[2] = '5';
                        Transiciones[3] = '6';
        }
        else if(Estados == 5 && Estado_sig == 3){
                        Transiciones[2] = '5';
                        Transiciones[3] = '3';
        }
        else if(Estados == 6 && Estado_sig == 1){
                        Transiciones[2] = '6';
                        Transiciones[3] = '1';
        }
        else if(Estados == 6 && Estado_sig == 5){
                        Transiciones[2] = '6';
                        Transiciones[3] = '5';
        }
        else{
                Transiciones[0] = '-';
                Transiciones[1] = '-';
                Transiciones[2] = '-';
                Transiciones[3] = '-';
                Estado_sig = Estados;
        }
        Transiciones[4] = 0;
}

void Rutina_Estados(void){
        if(Estados == 1){               //Estado Stop
                EValvula = 0;
                VF_estado = 0;
                VF_ref = 0;
                EKgen = 0;
                EKred = 0;
                EKload = 0;
                EFsinc = 0;
                EFrec = 0;
        }
        else if(Estados == 2){  //Estado Stand-by
                EValvula = EValvula_ch;
                VF_estado = 1;
                VF_ref = 30;
                if (Fe_PLLgen_f >= 30 && Fe_PLLgen_f<=49)
                        VF_ref = 9.5*Fe_PLLgen_f - 255;
                if(Fe_PLLgen_f >= 49)
                        VF_ref = 220;
                EKgen = 0;
                EKred = 0;
                EKload = 0;
                EFsinc = 0;
                if (Vgenrms >= 30  && Fe_PLLgen_f > 25){
                        EFrec = 1;
                }
        }
        else if(Estados == 3){  //Estado Sincronizado
                EValvula = EValvula_ch;
                VF_estado = 2;
                PF_ref = PF_ch;
                Phi_ref = Phi_ch;
                Phir_ref = Phir_ch;
                EKgen = 1;
                EKred = 1;
```

```
                EKload = 1;
                EFsinc = 1;
                EFrec = 1;
        }
        else if(Estados == 4){  //Estado Isla
                EValvula = EValvula_ch;
                VF_estado = 1;
                VF_ref = 220;
                EKgen = 1;
                EKred = 0;
                EKload = 1;
                EFsinc = 0;
                EFrec = 1;
        }
        else if(Estados == 5){  //Estado Red
                EValvula = EValvula_ch;
                VF_estado = 1;
                VF_ref = 30;
                if (Fe_PLLgen_f >= 30 && Fe_PLLgen_f<=49)
                        VF_ref = 9.5*Fe_PLLgen_f - 255;
                if(Fe_PLLgen_f >= 49)
                        VF_ref = 220;
                EKgen = 0;
                EKred = 1;
                EKload = 1;
                EFsinc = 0;
                if (Vgenrms >= 30  && Fe_PLLgen_f > 25){
                        EFrec = 1;
                }
        }
        else if(Estados == 6){  //Estado Mantencion
                VF_estado = 0;
                VF_ref = 0;
                EKgen = 0;
                EKred = 1;
                EKload = 1;
                EValvula = 0;
                EFsinc = 0;
                EFrec = 0;
        }
}

void Rutina_Transiciones(void){
        int tran = 1;
        if(strncmp(Transiciones,"TN12",4) == 0){
                Estados = 2;
                tran = 0;
        }
        else if(strncmp(Transiciones,"TN15",4) == 0){
                EKred = 1;
                EKload = 1;
                //if (verificar cierre de contactores){
                        Estados = 5;
                        tran = 0;
                //}
        }
        else if(strncmp(Transiciones,"TN16",4) == 0){
                EKred = 1;
                EKload = 1;
                //if (verificar cierre de contactores){
                        Estados = 6;
                        tran = 0;
                //}
                EFsinc = 0;
                EFrec = 0;
        }
        else if(strncmp(Transiciones,"TN21",4) == 0){
                EValvula = 0;
                VF_estado = 0;
```

```
                EFsinc = 0;
                EFrec = 0;
                //if (evaluar condicion de fin de carreras){
                        Estados = 1;
                        tran = 0;
                //}

        }
        else if(strncmp(Transiciones,"TN23",4) == 0){
                if(Vredrms >= 210 && Fe_PLLred_f >= 49.5 && Fe_PLLred_f <= 50.5){ // verificar
condiciones de la red
                        VF_estado = 1;
                        VF_ref = Vredrms-5;
                        EKred = 1;
                        if((abs(VF_ref-Vgenrms) < 3) && (abs(Fe_PLLgen_f-Fe_PLLred_f) <=0.1)){
//verificar tension y frecuencia del generador
                                        VFcnt++;
                                        if (VFcnt > 1000)
                                                VFcnt = 1000;
                                        //verificar cierre del contactor
                                        if((abs(Theta_PLLred-Theta_PLLgen)<= 0.03) && VFcnt >=
800){ // verificar fase del generador
                                                Theta_cnt++;
                                                if (Theta_cnt > 200){
                                                        EKgen = 1;
                                                        //if(verificar cierre del contactor
kgen){ // no implementada
                                                                EKload = 1;
                                                                Estados = 3;
                                                                PF_cnt = 50;
                                                                tran = 0;
                                                        //}
                                                }
                                        }
                                        else{
                                                Theta_cnt=0;
                                        }
                        }else{
                                VFcnt = 0;
                                Theta_cnt = 0;
                        }
                }
                else{                   // red no se encuentra en condiciones lleva a modo isla
                        EKred = 0;
                        VF_ref = 220;
                        Transiciones[0] = 'T';
                        Transiciones[1] = 'N';
                        Transiciones[2] = '2';
                        Transiciones[3] = '4';
                        Transiciones[4] = 0;
                }
        }
        else if(strncmp(Transiciones,"TN24",4) == 0){
                VF_estado = 1;
                VF_ref = 220;
                if(Vgenrms >= 210 && Fe_PLLgen_f > 49.5 && Fe_PLLgen_f < 50.5){
                        EKgen = 1;
                        EKload = 1;
                        //if(verificar cierre Kgen y Kload){
                                Estados = 4;
                                tran = 0;
                        //}
                }
                else{
                        Estados = 2;
                        tran = 0;
                }
        }
        else if(strncmp(Transiciones,"TN25",4) == 0){
```

```
                if(Vredrms >= 210 && Fe_PLLred_f > 49.5 && Fe_PLLred_f < 50.5){
                        EKred = 1;
                        EKload = 1;
                        //if(verificar cierre Kred y Kload){
                                Estados = 5;
                                tran = 0;
                        //}
                }
                else{
                        Estados = 2;
                        tran = 0;
                }
        }
        else if(strncmp(Transiciones,"TN26",4) == 0){
                EValvula = 0;
                VF_estado = 0;
                EFsinc = 0;
                EFrec = 0;
                if(Vredrms >= 210 && Fe_PLLred_f > 49.5 && Fe_PLLred_f < 50.5){
                        EKred = 1;
                        EKload = 1;
                        //if(verificar cierre Kred y Kload){
                                Estados = 6;
                                tran = 0;
                        //}
                }
                else{
                        Estados = 2;
                        tran = 0;
                }
        }
        else if(strncmp(Transiciones,"TN41",4) == 0){
                EKgen = 0;
                EKload = 0;
                EFsinc = 0;
                EFrec = 0;
                //if (verificar apertura kgen y kload){
                        EValvula = 0;
                        VF_estado = 0;
                        tran = 0;
                        Estados = 1;
                //}
        }
        else if(strncmp(Transiciones,"TN42",4) == 0){
                EKgen = 0;
                EKload = 0;
                //if (verificar apertura kgen y kload){
                        tran = 0;
                        Estados = 2;
                //}
        }
        else if(strncmp(Transiciones,"TN43",4) == 0){
                if (Cancel == 1){
                        tran = 0;
                        Estado_sig = Estados;
                }
                if(Vredrms >= 210 && Fe_PLLred_f >= 49.5 && Fe_PLLred_f <= 50.5){
                        VF_estado = 1;
                        VF_ref = Vredrms-5;
                        if((abs(VF_ref-Vgenrms) < 3) && (abs(Fe_PLLgen_f-Fe_PLLred_f) <=0.1)){
                                        VFcnt++;
                                        if (VFcnt > 1000)
                                                VFcnt = 1000;
                                        //verificar cierre del contactor
                                        if((abs(Theta_PLLred-Theta_PLLgen)<= 0.03) && VFcnt >=
800){
                                                Theta_cnt++;
                                                if (Theta_cnt > 200){
                                                        EKred = 1;
```

```
                                                        //if(verificar cierre del contactor
kred){
                                                                Estados = 3;
                                                                tran = 0;
                                                                PF_cnt = 50;
                                                        //}
                                                }
                                        }
                                        else{
                                                Theta_cnt=0;
                                        }
                        }else{
                                VFcnt = 0;
                                Theta_cnt = 0;
                        }
                }
                else{
                        tran = 0;
                        Estados = 4;
                }
        }
        else if(strncmp(Transiciones,"TN31",4) == 0){
                EKgen = 0;
                //if (verificar apertura kgen){
                        EValvula = 0;
                        VF_estado = 0;
                        EKload = 0;
                        EKred = 0;
                        EFsinc = 0;
                        EFrec = 0;
                        //if (verificar apertura kred y kload){
                                tran = 0;
                                Estados = 1;
                        //}
                //}
        }
        else if(strncmp(Transiciones,"TN32",4) == 0){
                EKgen = 0;
                //if (verificar apertura kgen){
                        VF_estado = 1;
                        VF_ref = 220;
                        EKred = 0;
                        EKload = 0;
                        //if (verificar apertura kgen){
                                tran = 0;
                                Estados = 2;
                        //}
                //}
        }
        else if(strncmp(Transiciones,"TN34",4) == 0){
                EKred = 0;
                //if (verificar apertura kred){
                        VF_estado = 1;
                        VF_ref = 220;
                        Estados = 4;
                        tran = 0;
                //}
        }
        else if(strncmp(Transiciones,"TN35",4) == 0){
                EKgen = 0;
                //if (verificar apertura kgen){
                        VF_estado = 1;
                        VF_ref = 220;
                        tran = 0;
                        Estados = 5;
                //}
        }
        else if(strncmp(Transiciones,"TN36",4) == 0){
                EKgen = 0;
```

```
		//if (verificar apertura kgen){
			EValvula = 0;
			VF_estado = 0;
			EFsinc = 0;
			EFrec = 0;
			//if(verificar fin de carrera valvula){
				tran = 0;
				Estados = 6;
			//}
		//}
	}
	else if(strncmp(Transiciones,"TN51",4) == 0){
		EValvula = 0;
		VF_estado = 0;
		//if (verificar FIN DE CARRERA VALVULA){
			EKload = 0;
			EKred = 0;
			EFsinc = 0;
			EFrec = 0;
			//if (verificar apertura kred y kload){
				tran = 0;
				Estados = 1;
			//}
		//}
	}
	else if(strncmp(Transiciones,"TN56",4) == 0){
		EValvula = 0;
		VF_estado = 0;
		EFsinc = 0;
		EFrec = 0;
		//if (verificar FIN DE CARRERA VALVULA){
			tran = 0;
			Estados = 6;
		//}
	}
	else if(strncmp(Transiciones,"TN53",4) == 0){
		VF_estado = 1;
		VF_ref = Vredrms-5;
		if((abs(VF_ref-Vgenrms) < 3) && (abs(Fe_PLLgen_f-Fe_PLLred_f) <=0.1)){
				VFcnt++;
				if (VFcnt > 1000)
					VFcnt = 1000;
				//verificar cierre del contactor
				if((abs(Theta_PLLred-Theta_PLLgen)<= 0.03) && VFcnt >= 800){
					Theta_cnt++;
					if (Theta_cnt > 200){
						EKgen = 1;
						//if(verificar cierre del contactor kgen){
							Estados = 3;
							PF_cnt = 50;
							tran = 0;
						//}
					}
				}
				else{
					Theta_cnt=0;
				}
		}else{
			VFcnt = 0;
			Theta_cnt = 0;
		}
	}
	else if(strncmp(Transiciones,"TN61",4) == 0){
		EKload = 0;
		EKred = 0;
		//if (verificar apertura kred y kload){
			tran = 0;
			Estados = 1;
		//}
```

```
        }
        else if(strncmp(Transiciones,"TN65",4) == 0){
                VF_estado = 1;
                Estados = 5;
                tran = 0;
        }

        if (tran == 0){
                Transiciones[0] = '-';
                Transiciones[1] = '-';
                Transiciones[2] = '-';
                Transiciones[3] = '-';
                Transiciones[4] = 0;
        }
}

void Rutina_Control(void){
        if(VF_estado == 1){             // Control de Tension
                Duty_max = (int) (EPWM1_TBPRD_05K*0.59);
                Vf_max   =      Duty_max*KDuty;

                errVgen = VF_ref-Vgenrms;
                IntVgen1=IntVgen0+ KIVgen*errVgen;
                Vf=IntVgen1+KPVgen*errVgen;

                if(Vf > Vf_max) Vf = Vf_max;
                else if(Vf < Vf_min) Vf = Vf_min;
                else IntVgen0=IntVgen1;

        }
        else if(VF_estado == 2){// Control de FP
                Duty_max = (int) (EPWM1_TBPRD_05K*Duty_max_ref);
                Vf_max = Duty_max*KDuty;

                if(PF_cnt >= 50){

                        errPFgen = Phir - Phir_ref;
                        IntPFgen1 = IntPFgen0 + KIPFgen*errPFgen;
                        ProPFgen1 = KPPFgen*errPFgen;
                        Vf = IntPFgen1 + ProPFgen1;

                        if(Vf > Vf_max2) Vf = Vf_max2;
                        else if(Vf < Vf_min2) Vf = Vf_min2;
                        else IntPFgen0 = IntPFgen1;

                        PF_cnt = 0;
                }
                else
                        PF_cnt++;
        }
        else{
                VF_estado = 0;
                Vf = Vf_min;
        }

        Duty = (int) (Vf/KDuty);
        if(Duty > Duty_max) Duty = Duty_max;
        else if(Duty < Duty_min) Duty = Duty_min;

        Changeduty(Duty);
}



void Rutina_Actuador(void){
        if(EKgen == 1) KGEN_CERRAR
        else KGEN_ABRIR
        if(EKred == 1) KRED_CERRAR
        else KRED_ABRIR
```

```
        if(EFsinc == 1) CONTROL_SINC_ON
        else CONTROL_SINC_OFF
        if(EFrec == 1)CONTROL_RECT_ON
        else CONTROL_RECT_OFF
}

interrupt void cpu_timer0_isr(void){
        timer_isr_cnt++;
        if (timer_isr_cnt>10){
                GpioDataRegs.GPATOGGLE.bit.GPIO12 = 1;
                GpioDataRegs.GPATOGGLE.bit.GPIO31 = 1;
                timer_isr_cnt = 0;
        }

        //Estado_falla; Estado_TX; id_TX;


        SciaRegs.SCIFFTX.bit.TXFIFOXRESET =1;  // enable TXFIFO
        SciaRegs.SCIFFTX.bit.TXFFINTCLR = 1 ;  // force TX-ISR
        if(id_TX==1){
                build_msgS();
                if(Estado_falla!=0) id_TX = 4;
                if(Estado_TX>1 && Estado_TX<4) id_TX = Estado_TX;
                else{Estado_TX=1; id_TX = 1;}
        }
        else if(id_TX==2){
                build_msgG();
                id_TX = 1;
                if(Estado_falla!=0) id_TX = 4;
        }
        else if(id_TX==3){
                build_msgR();
                id_TX = 1;
                if(Estado_falla!=0) id_TX = 4;
        }
        else{
                build_msgFA();
                id_TX = 1;
        }


        EALLOW;
        SysCtrlRegs.WDKEY = 0x55;               // Service watchdog #1
        EDIS;

        PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;

}

void build_msgS(void){                                                  //State Msg
        msg[0]='S';
        msg[1]= (((int)Vgenrms) & 0xFF00) >> 8;         //Vgen
        msg[2]= ((int)Vgenrms) & 0x00FF;
        msg[3]= (((int)Vredrms) & 0xFF00) >> 8;         //Vred
        msg[4]= ((int)Vredrms) & 0x00FF;
        msg[5]= (((int) abs(Pgen)) & 0xFF00) >> 8;      //Pgen
        msg[6]= ((int) abs(Pgen)) & 0x00FF;
        msg[7]= (((int) abs(Pred)) & 0xFF00) >> 8;      //Pred
        msg[8]= ((int) abs(Pred)) & 0x00FF;
        msg[9]= Estados & 0x00FF;                                               //Estado
        msg[10]= Estado_sig & 0x00FF;                           //Estado siguiente
        msg[11]= EValvula & 0x00FF;                                             //Valvula
        msg[12]= (EKgen+EKred*2+EKload*4) & 0x00FF;     //Contactores
        msg[13]='\n';
        msg[14]='\r';
}
```

```
void build_msgG(void){

        if(Estados==3){
                VgenA_rms = VredA_rms;
                VgenB_rms = VredB_rms;
                VgenC_rms = VredC_rms;
        }

        int auxPA=VgenA_rms*IgenA_rms;
        int auxPB=VgenB_rms*IgenB_rms;
        int auxPC=VgenC_rms*IgenC_rms;

        msg[0]='G';
        msg[1]= (((int)VgenA_rms) & 0xFF00) >> 8;       // VA
        msg[2]= ((int)VgenA_rms) & 0x00FF;
        msg[3]= (((int)VgenB_rms) & 0xFF00) >> 8;       // VB
        msg[4]= ((int)VgenB_rms) & 0x00FF;
        msg[5]= (((int)VgenC_rms) & 0xFF00) >> 8;       // VC
        msg[6]= ((int)VgenC_rms) & 0x00FF;
        msg[7]= (auxPA & 0xFF00) >> 8;  // IA
        msg[8]= auxPA & 0x00FF;
        msg[9]= (auxPB & 0xFF00) >> 8;  // IB
        msg[10]= auxPB & 0x00FF;
        msg[11]= (auxPC & 0xFF00) >> 8; // IC
        msg[12]= auxPC & 0x00FF;
        msg[13]='\n';
        msg[14]='\r';
}

void build_msgR(void){

        int auxPA=VredA_rms*IredA_rms;
        int auxPB=VredB_rms*IredB_rms;
        int auxPC=VredC_rms*IredC_rms;

        msg[0]='R';
        msg[1]= (((int)VredA_rms) & 0xFF00) >> 8;       // VA
        msg[2]= ((int)VredA_rms) & 0x00FF;
        msg[3]= (((int)VredB_rms) & 0xFF00) >> 8;       // VB
        msg[4]= ((int)VredB_rms) & 0x00FF;
        msg[5]= (((int)VredC_rms) & 0xFF00) >> 8;       // VC
        msg[6]= ((int)VredC_rms) & 0x00FF;
        msg[7]= (auxPA & 0xFF00) >> 8;  // IA
        msg[8]= auxPA & 0x00FF;
        msg[9]= (auxPB & 0xFF00) >> 8;  // IB
        msg[10]= auxPB & 0x00FF;
        msg[11]= (auxPC & 0xFF00) >> 8; // IC
        msg[12]= auxPC & 0x00FF;
        msg[13]='\n';
        msg[14]='\r';
}

void build_msgFA(void){ // Falla
        msg[0]='A';
        msg[1]='A';
        msg[2]='A';
        msg[3]='A';
        msg[4]='A';
        msg[5]='A';
        msg[6]='A';
        msg[7]='A';
        msg[8]='A';
        msg[9]='A';
        msg[10]='A';
        msg[11]='A';
        msg[12]='A';
        msg[13]='\n';
        msg[14]='\r';
```

```
}

interrupt void SCIA_RX_isr(void){
        int i;
        for (i=0;i<16;i++) Brx[i]= SciaRegs.SCIRXBUF.bit.RXDT;
        leer_rx();
        SciaRegs.SCIFFRX.bit.RXFIFORESET = 0;    // reset RX-FIFO pointer
        SciaRegs.SCIFFRX.bit.RXFIFORESET = 1;    // enable RX-operation
        SciaRegs.SCIFFRX.bit.RXFFINTCLR = 1;     // clear RX-FIFO INT Flag
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;
}

interrupt void SCIA_TX_isr(void){
        unsigned int i;
        for(i=0;i<14;i++) SciaRegs.SCITXBUF= msg[i];
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;
}

void leer_rx(void){
        int aux;
        if (Brx[0]=='E' && Brx[4]=='\n'){
                aux = ((Brx[1] & 0x00FF) << 8) + (Brx[2] & 0x00FF);
                if(aux>0 && aux<7)      Estado_sig = aux;
        }
        if (Brx[0]=='V' && Brx[4]=='\n'){
                aux = ((Brx[1] & 0x00FF) << 8) + (Brx[2] & 0x00FF);
                if (aux >= 0)   VF_ch = (float) aux;
                Ten = aux;
        }
        if (Brx[0]=='P' && Brx[4]=='\n'){
                aux = ((Brx[1] & 0x00FF) << 8) + (Brx[2] & 0x00FF);
                if (aux >= 0 && aux <= 100)     PF_ch = ((float)aux)*0.01;
        }
        if (Brx[0]=='T' && Brx[4]=='\n'){
                aux = ((Brx[1] & 0x00FF) << 8) + (Brx[2] & 0x00FF);
                if(aux >= 0 && aux <= 10){      Estado_TX = aux; id_TX = 0;}
        }
        if (Brx[0]=='A' && Brx[4]=='\n'){
                aux = ((Brx[1] & 0x00FF) << 8) + (Brx[2] & 0x00FF);
                if(aux >= 0 && aux <= 100){     EValvula_ch = aux;}
        }
}



void Changeduty(int dutycycle){
        EPwm1Regs.CMPB  = dutycycle;
}

//###########################################################################
// FIN
//###########################################################################
```

A.2. Configuración de los ADC

```
#include "DSP2833x_Device.h"     // DSP2833x Header file Include File

/*
 *ADCTRL1, ADCTRL2 y ADCTRL3 register set up
 *
 * SEQUENCER MODE: Selects whether we use state machine of Auto Sequencer as a single 16 stage
state machine ("Cascade Mode")
 * or as a pair of two independent 8-stage measurement unit ("Dual sequencer Mode").
 *
 * SAMPLING MODE: Convert 2 analog input signals at one time in "Simultaneous Sampling". If we
choose "Sequential Sampling"
 * only one multiplexed input channel is converted at one time.
 *
 * START MODE: Selecting "Single Sequence Mode" (or "Start/Halt - Mode") the auto sequencer
starts at the first input trigger
 * signal, performs the predefined number of conversions and stops at the end.
 *
 */

void ConfigADCregister(void)
{

        AdcRegs.ADCTRL1.all = 0;                // Set ADCTRL1 in Zero
        /*
                 bit 15                 x: Reserved     (Write no have effect)
                 bit 14                 0: RESET                Reset.
                0 = no effect, 1 = Reset entire module (bit is then set back to 0 by ADC logic)
                 bit 13-12     00: SUSMOD                Emulation suspend Mode          00 =
Emulation suspend is ignored
                 bit 11-8     111: ACQ_PS               Acquisition window size Acquisition time
= (ADCTRL[11:8]+1)*times the ADCLK period
                 bit 7          0: CPS                  Core Clock Prescaler            0 =
ADCCLK=F_clk/1,  1 = ADCCLK=F_clk/2
                 bit 6                  0: CONT_RUN     Continuous run
        0 = Stop after reaching end of sequence, 1 = Continuous (start all over again from
"initial state")
                 bit 5                  0: SEQ_OVRD             Sequencer Override
        0 = Sequencer pointer reset to "initial state" at end of MAX_CONVn, 1 = Sequencer resets
to "initial state" after "end state"
                 bit 4                  1: SEQ_CASC             Sequencer Mode
        0 = Dual Mode, 1 = Cascade Mode
                 bit 3-0 xxxx: Reserved         (Write no have effect)
        */
        AdcRegs.ADCTRL1.bit.ACQ_PS              = 0xf;//0x7
        AdcRegs.ADCTRL1.bit.CPS         = 0;
        AdcRegs.ADCTRL1.bit.CONT_RUN    = 0;
        AdcRegs.ADCTRL1.bit.SEQ_CASC    = 1;



        AdcRegs.ADCTRL2.all = 0;
        /*
            bit 15                      0: ePWM_SOCB_SEQePWM SOCB enable for cascade sequencer
        0 = No action, 1 = Allows the cascade seq to be started by an ePWM SOCB
            bit 14                      0: RST_SEQ1                         Reset sequencer 1
                                                0 = No action, 1 = Inmediate reset SEQ1 to
"initial state"
            bit 13          0: SOC_SEQ1                 SOC trigger for seq 1
                        0 = Clear a pending SOC trigger, 1 = Software trigger - Satart SEQ1 from
currently stopped position
            bit 12          x: Reserved
            bit 11          1: INT_ENA_SEQ1             SEQ1 interrupt enable
                        0 = Interrupt request by INT_SEQ1 is disabled, 1 = Interrupt request by
INT_SEQ1 is enabled
```

```
            bit 10                      0: INT_MOD_SEQ1         SEQ1 interrupt mode
                                        0 = INT_SEQ1 is set at the end of every SEQ1 sequence, 1
= INT_SEQ1 is set at the end of every other SEQ1 seq
            bit 9                       x: Reserved
            bit 8                       1: ePWM_SOCA_SEQ1       ePWM SOCA enable bit for SEQ1
                        0 = SEQ1 cannot be started by ePWMx SOCA trigger, 1 = Allows SEQ1/SEQ to
be started by ePWMx SOCA trigger
            bit 7                       0: EXT_SOC_SEQ1         External signal SOC for SEQ1
                        0 = No action, 1 = Enable ADC autoconversion seq by a signal from GPIO
port A, pin GPIO31-0
            bit 6                       0: RST_SEQ2                     Reset Sequencer 2
                                                0 = No action, 1 = Inmediate reset SEQ2 to
"initial state"
            bit 5                       0: SOC_SEQ2                     SOC trigger for seq 2
                                        0 = Clear a pending SOC trigger, 1 = Software trigger -
Satart SEQ2 from currently stopped position
            bit 4                       0: Reserved
            bit 3                       0: INT_ENA_SEQ2         SEQ2 interrupt enable
                                0 = Interrupt request by INT_SEQ2 is disabled, 1 = Interrupt
request by INT_SEQ2 is enabled
            bit 2                       0: INT_MOD_SEQ2         SEQ2 interrupt mode
                                        0 = INT_SEQ2 is set at the end of every SEQ2 sequence, 1
= INT_SEQ2 is set at the end of every other SEQ2 seq
            bit 1                       0: Reserved
            bit 0                       0: ePWM_SOCB_SEQ2       ePWM SOCA enable bit for SEQ2
                        0 = SEQ2 cannot be started by ePWMx SOCB trigger, 1 = Allows SEQ2 to be
started by ePWMx SOCB trigger
        */
        AdcRegs.ADCTRL2.bit.INT_ENA_SEQ1        = 1;
        AdcRegs.ADCTRL2.bit.EPWM_SOCA_SEQ1      = 1;
        AdcRegs.ADCTRL2.bit.INT_MOD_SEQ1        = 0;
        AdcRegs.ADCTRL2.bit.RST_SEQ1 = 0x1;

        AdcRegs.ADCTRL3.bit.ADCCLKPS = 1;
        /*
            bit 15-8    xxxxxxxx: Reserved
            bit 7-6 00: ADCBGRFDN ADC Bandgap and Reference Power Down 00 = Powered down,
                    11 = Powered up
            bit 5 0: ADCPWDN  ADC power Down 0 = Powered down,  1 = Powered up
            bit 4-1 0000: ADCCLKPS  ADC Clock Prescale
                      0 = FCLK=HSPCLK, 1 to F = FCLK=HSPCLK/(2*ADCCLKPS)
            bit 0      0: SMODE_SEL Sampling Mode Select
0 = Sequential sampling mode, 1 = Simultaneous sampling mode
        */
        AdcRegs.ADCMAXCONV.all = 15;
/*
  bit 15-7   xxxxxxxxx: Reserved
  bit 6-4 000: MAX_CONV2  Define the maximum number of conversion executed en autoconversion,
               For SEQ2 operation bits MAX_CONV2[2:0] are used
  bit 3-0 000: MAX_CONV1 For SEQ1 operation bits MAX_CONV1[2:0] are used and For SEQ operation
               bits MAX_CONV1[3:0] are used
  */
        AdcRegs.ADCCHSELSEQ1.bit.CONV00 = 0; // Setup ADCINA0 as 1st SEQ1 conv.
        AdcRegs.ADCCHSELSEQ1.bit.CONV01 = 0x8;
        AdcRegs.ADCCHSELSEQ1.bit.CONV02 = 0x1;
        AdcRegs.ADCCHSELSEQ1.bit.CONV03 = 0x9;
        AdcRegs.ADCCHSELSEQ2.bit.CONV04 = 0x2;
        AdcRegs.ADCCHSELSEQ2.bit.CONV05 = 0xa;
        AdcRegs.ADCCHSELSEQ2.bit.CONV06 = 0x3;
        AdcRegs.ADCCHSELSEQ2.bit.CONV07 = 0xb;
        AdcRegs.ADCCHSELSEQ3.bit.CONV08 = 0x4;
        AdcRegs.ADCCHSELSEQ3.bit.CONV09 = 0xc;
        AdcRegs.ADCCHSELSEQ3.bit.CONV10 = 0x5;
        AdcRegs.ADCCHSELSEQ3.bit.CONV11 = 0xd;
        AdcRegs.ADCCHSELSEQ4.bit.CONV12 = 0x6;
        AdcRegs.ADCCHSELSEQ4.bit.CONV13 = 0xe;
        AdcRegs.ADCCHSELSEQ4.bit.CONV14 = 0x7;
        AdcRegs.ADCCHSELSEQ4.bit.CONV15 = 0xf;
```

```
        /*
        ADCCHELSEQ1
                bit 15-12 0x9: CONV03
                bit 11-8  0x1: CONV02
                bit 7-4   0x8: CONV01
                bit 3-0   0x0: CONV00

        ADCCHELSEQ2
                bit 15-12 0xb: CONV03
                bit 11-8  0x3: CONV02
                bit 7-4   0xa: CONV01
                bit 3-0   0x2: CONV00

        ADCCHELSEQ3
                bit 15-12 0xd: CONV03
                bit 11-8  0x5: CONV02
                bit 7-4   0xc: CONV01
                bit 3-0   0x4: CONV00

        ADCCHELSEQ4
                bit 15-1  0xf: CONV03
                bit 11-8  0x7: CONV02
                bit 7-4   0xe: CONV01
                bit 3-0   0x6: CONV00


        CONVnn Value    ADC input channel selected
                0000    ADCINA0
                0001    ADCINA1
                0010    ADCINA2
                0011    ADCINA3
                0100    ADCINA4
                0101    ADCINA5
                0110    ADCINA6
                0111    ADCINA7
                1000    ADCINB0
                1001    ADCINB1
                1010    ADCINB2
                1011    ADCINB3
                1100    ADCINB4
                1101    ADCINB5
                1110    ADCINB6
                1111    ADCINB7
        */

}
```

A.3. Configuración de los GPIOs

```
#include "DSP2833x_Device.h"     // DSP2833x Header file Include File

void ConfigGPIOregister(void){

        EALLOW;

        //General configuration as general purpose I/O

                GpioCtrlRegs.GPAMUX1.all = 0;
         /*
            bit 31-0  0: GPIO 15 ... GPIO 0     Configure GPIO like general purpose I/O
         */
                GpioCtrlRegs.GPAMUX2.all = 0;
```

```
        /*
          bit 31-0      0: GPIO 31 ... GPIO 16  Configure GPIO like general purpose I/O
        */
                GpioCtrlRegs.GPBMUX1.all = 0;   // GPIO47 ... GPIO32 = General Purpose I/O
                GpioCtrlRegs.GPBMUX2.all = 0;   // GPIO63 ... GPIO48 = General Purpose I/O
                GpioCtrlRegs.GPCMUX1.all = 0;   // GPIO79 ... GPIO64 = General Purpose I/O
                GpioCtrlRegs.GPCMUX2.all = 0;   // GPIO87 ... GPIO80 = General Purpose I/O

                //GpioCtrlRegs.GPAMUX2.bit.GPIO31 = 0;  //led
                //GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0;  //led


                //Particular configuration

                GpioCtrlRegs.GPAMUX2.bit.GPIO28 = 1;
                GpioCtrlRegs.GPAMUX2.bit.GPIO29 = 1;
                        /*
                          bit 27-26             01: GPIO 29     Configure GPIO like SCITXDA
                          bit 25-24             01: GPIO 28     Configure GPIO like SCIRXDA
                        */

                GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 1;     // PWM Campo
        //      GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 1;     // PWM Resistencia  desahogo fase A
        //      GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 1;     // PWM Resistencia  desahogo fase B
        //      GpioCtrlRegs.GPAMUX1.bit.GPIO7 = 1;     // PWM Resistencia  desahogo fase C


                //Define as input or output

                GpioCtrlRegs.GPADIR.all = 0;
        /*
        bit 31-0  0: GPIO 31-0  0 = Configure GPIO as input     1 = Configure GPIO as output
        */
                GpioCtrlRegs.GPADIR.bit.GPIO0 = 1;      // output Reles 220 inversor
                GpioCtrlRegs.GPAPUD.bit.GPIO0 = 1;      // 0 = Pull up on, 1 = Pull up off
                GpioDataRegs.GPASET.bit.GPIO0 = 1;      // 0 = ignored, 1 = output high

                GpioCtrlRegs.GPADIR.bit.GPIO1 = 1;      // output PWM1B Campo

                GpioCtrlRegs.GPADIR.bit.GPIO2 = 1;      // output PWM Extra
                GpioCtrlRegs.GPAPUD.bit.GPIO2 = 1;      // 0 = Pull up on, 1 = Pull up off
                GpioDataRegs.GPASET.bit.GPIO2 = 1;      // 0 = ignored, 1 = output high

                GpioCtrlRegs.GPADIR.bit.GPIO3 = 1;  // output PWM2B Resistencia  desahogo fase A

                GpioCtrlRegs.GPADIR.bit.GPIO4 = 1;      // GPIO
                GpioCtrlRegs.GPAPUD.bit.GPIO4 = 1;      // 0 = Pull up on, 1 = Pull up off
                GpioDataRegs.GPASET.bit.GPIO4 = 1;      // 0 = ignored, 1 = output high

                GpioCtrlRegs.GPADIR.bit.GPIO5 = 1;  // output PWM3B Resistencia  desahogo fase B

                GpioCtrlRegs.GPADIR.bit.GPIO6 = 1;      // Control de frecuencia
                GpioCtrlRegs.GPAPUD.bit.GPIO6 = 1;      // 0 = Pull up on, 1 = Pull up off
                GpioDataRegs.GPASET.bit.GPIO6 = 1;      // 0 = ignored, 1 = output high

                GpioCtrlRegs.GPADIR.bit.GPIO7 = 1;  // output PWM4B Resistencia  desahogo fase C

                GpioCtrlRegs.GPADIR.bit.GPIO8 = 1;      // output Reles Motor Valvula
                GpioCtrlRegs.GPAPUD.bit.GPIO8 = 1;      // 0 = Pull up on, 1 = Pull up off
                GpioDataRegs.GPASET.bit.GPIO8 = 1;      // 0 = ignored, 1 = output pin high

                GpioCtrlRegs.GPADIR.bit.GPIO10 = 1;     // output GPIO extra
                GpioCtrlRegs.GPADIR.bit.GPIO12 = 1;     // output GPIO extra
                GpioCtrlRegs.GPADIR.bit.GPIO15 = 1;     // output GPIO Led 1
                GpioCtrlRegs.GPADIR.bit.GPIO16 = 1;     // output GPIO Led 3

                GpioCtrlRegs.GPADIR.bit.GPIO25 = 0;     // in GPIO Fin carrera
```

```
                GpioCtrlRegs.GPADIR.bit.GPIO26 = 1;     // output GPIO Led 2
                GpioCtrlRegs.GPADIR.bit.GPIO27 = 0;     // in GPIO Fin carrera
                GpioCtrlRegs.GPADIR.bit.GPIO31 = 1;     // output LED DSC

                GpioCtrlRegs.GPBDIR.all = 0;            // GPIO63-32 as inputs
                GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1;     // output LED DSC

                GpioCtrlRegs.GPBDIR.bit.GPIO48 = 1;     // output GPIO extra abrir cerrar
valvula
                GpioCtrlRegs.GPBPUD.bit.GPIO48 = 1;     // 0 = Pull up on, 1 = Pull up off
                GpioDataRegs.GPBSET.bit.GPIO48 = 1;     // 0 = ignored, 1 = output pin high

                GpioCtrlRegs.GPBDIR.bit.GPIO59 = 1;     // output Reles alimentacion inversor
                GpioCtrlRegs.GPBPUD.bit.GPIO59 = 1;     // 0 = Pull up on, 1 = Pull up off
                GpioDataRegs.GPBSET.bit.GPIO59 = 1;     // 0 = ignored, 1 = output pin high

                GpioCtrlRegs.GPBDIR.bit.GPIO58 = 1;     // output Reles contactor generador
                GpioCtrlRegs.GPBPUD.bit.GPIO58 = 1;     // 0 = Pull up on, 1 = Pull up off
                GpioDataRegs.GPBSET.bit.GPIO58 = 1;     // 0 = ignored, 1 = output pin high

                GpioCtrlRegs.GPBDIR.bit.GPIO60 = 1;     // output Reles contactor red
                GpioCtrlRegs.GPBPUD.bit.GPIO60 = 1;     // 0 = Pull up on, 1 = Pull up off
                GpioDataRegs.GPBSET.bit.GPIO60 = 1;     // 0 = ignored, 1 = output pin high

                GpioCtrlRegs.GPBDIR.bit.GPIO61 = 1;     // output Reles 220 red
                GpioCtrlRegs.GPBPUD.bit.GPIO61 = 1;     // 0 = Pull up on, 1 = Pull up off
                GpioDataRegs.GPBSET.bit.GPIO61 = 1;     // 0 = ignored, 1 = output pin high

                GpioCtrlRegs.GPCDIR.all = 0;            // GPIO87-64 as inputs
                GpioCtrlRegs.GPCDIR.bit.GPIO84 = 1;     // output GPIO extra
                GpioCtrlRegs.GPCDIR.bit.GPIO86 = 1;     // output GPIO extra

                EDIS;




}
```

B. Programa de la HMI

B.1. Globals.h

```
#ifndef GLOBALS H
#define GLOBALS_H

#include "uwindow.h"
#include "ui_uwindow.h"
#include <QMutex>

#ifndef WINDOWS H
#define WINDOWS_H
#include <windows.h>
#endif

extern HANDLE hFile;
extern DCB dcbSerialParams;
extern COMMTIMEOUTS timeouts;
extern DWORD dwBytesRead;

typedef struct ElecVars{

    float Fgen;
    float Fred;
    float FP;
    int VABC[3];
    int PABC[3];
    int Est;
    int Est sig;
    int K[3];
}ElectVars;

extern QMutex MutexEVars;
extern ElectVars EVars;
void initEVars(ElectVars * e);

typedef struct MecVars{
    int val;
    int caudal;
    int nivel;
} MecVars;

extern QMutex MutexMVars;
extern MecVars MVars;
void initMVars(MecVars *m);

extern Ui::uWindow *wind;

typedef struct RefTx{
    int Est sig;
    int Estado_Tx;
} RefTx;

extern QMutex MutexRefVars;
extern RefTx RefVars;
void initRefVars(RefTx *r);

extern int Tx;
extern int Rx_err;
extern int Rx_tot;
extern int Rx_bue;

#endif // GLOBALS_H
```

B.2. Leerserial.h

```
#ifndef LEERSERIAL_H
#define LEERSERIAL H

#include <QObject>
#include <QDebug>

#ifndef MYGLOBALS H
#define MYGLOBALS_H
#include <globals.h>
#endif

class LeerSerial : public QObject
{
    Q_OBJECT
public:
    explicit LeerSerial(QObject *parent = 0);
    int unir(char a, char b);
    void deciferSmsg(char *data);
    void deciferGRmsg(char *data);
    void sendmsg();

signals:
    void SerialReadEnd();
    void sigACT();

public slots:
    void SerialRead();

};

#endif // LEERSERIAL_H
```

B.3. Serial.h

```
#ifndef SERIAL_H
#define SERIAL_H


#ifndef MYGLOBALS_H
#define MYGLOBALS H
#include <globals.h>
#endif

#include <QObject>
#include <QThread>
#include <QDebug>
#include "leerserial.h"

#ifndef MYTIMER_H
#define MYTIMER_H
#include <QTimer>
#endif


class Serial : public QObject
{
```

```
    Q_OBJECT
public:
    Serial(QObject *parent = 0);
    LeerSerial *lee;

signals:


public slots:
    bool SerialOpen();
    void SerialClose();
    void SerialTime();

private:
    QTimer *timer;
    QThread *thread;


};

#endif // SERIAL H
```

B.4. Uwindow.h

```
#ifndef UWINDOW_H
#define UWINDOW_H

#include <QMainWindow>
#include <stdlib.h>
#include <QDebug>
#include <QPushButton>
#include <QIcon>

#ifndef MYGLOBALS_H
#define MYGLOBALS_H
#include <globals.h>
#endif

namespace Ui {
class uWindow;
}

class uWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit uWindow(QWidget *parent = 0);
    ~uWindow();

private slots:
    void on_pbGCR_clicked();
    void on_pbMA_clicked();
    void on_pbG_clicked();
    void on_pbR_clicked();
    void on_pbAL_clicked();
    void on_pbHR_clicked();
    void on_pbRG_clicked();
    void on_pbAY_clicked();

    void on_pb_MA_M_clicked();
    void on_pb_MA_A_clicked();
    void on_pbDet_clicked();
```

```
    void on_pbEsp_clicked();
    void on_pbIsl_clicked();
    void on_pbSic_clicked();
    void on_pbRed_clicked();
    void on_pbMan_clicked();

    void Act();
    void ActSys(int e);
    void ActGCR();
    void ActGR();
    void ActMA();

signals:
    void sigTransmit();

private:
    Ui::uWindow *ui;
    QPushButton *button;
    QPushButton *buttonE;
    int MA;
    QIcon ReIcon;
};

#endif // UWINDOW_H
```

B.5. Globals.c

```
#include <globals.h>

HANDLE hFile;
DCB dcbSerialParams;
COMMTIMEOUTS timeouts;
DWORD dwBytesRead;

QMutex MutexEVars;
ElectVars EVars;

void initEVars(ElectVars * e){
   e->Fgen = 0;
   e->Fred = 0;
   e->FP = 1;
   for(int i =0;i<3;i++){
       e->VABC[i] = 0;
       e->PABC[i] = 0;
       e->K[i]=0;
   }
   e->Est = 1;
   e->Est_sig = 1;
}
QMutex MutexMVars;
MecVars MVars;

void initMVars(MecVars *m){
    m->caudal = 0;
    m->nivel = 100;
    m->val = 0;
}

Ui::uWindow *wind;

QMutex MutexRefVars;
RefTx RefVars;

void initRefVars(RefTx *r){
```

```
    r->Est_sig = 1;
    r->Estado_Tx = 1;
}

int Tx = 0;
int Rx_err = 0;
int Rx_tot = 0;
int Rx_bue = 0;
```

B.6. Leerserial.c

```
#include "leerserial.h"

LeerSerial::LeerSerial(QObject *parent) :
    QObject(parent)
{
}

void LeerSerial::SerialRead(){
    if(Tx == 0){
        char bufRx[15];
        //qDebug()<<"Read!!";
        if(!ReadFile(hFile, bufRx, 15, &dwBytesRead, NULL)){
        //error occurred. Report to user.
            qDebug()<<"read error"<<"\n";
        }
        //qDebug()<<bufRx[0];
        if(Rx_tot <1000)
            Rx_tot++;
        if(bufRx[0]=='S' && bufRx[13]=='\n')
            deciferSmsg(bufRx);
        else if(bufRx[0]=='G' && bufRx[13]=='\n')
            deciferGRmsg(bufRx);
        else if(bufRx[0]=='R' && bufRx[13]=='\n')
            deciferGRmsg(bufRx);
        else{
            if(Rx_tot <=999)
                Rx err++;
            //qDebug()<<"error en Rx: "<<Rx_err;
            //qDebug()<<"buf: "<<bufRx;
            //qDebug()<<"buf 0: "<<bufRx[0];
            //qDebug()<<"buf 13: "<<((int) bufRx[13]);
        }
        qDebug()<<"tot: "<<Rx tot<<" err: "<<Rx err<<" buenos: "<<Rx bue;
        emit sigACT();
    }
    else{
        sendmsg();
        Tx = 0;
    }
    emit sigACT();
    emit SerialReadEnd();
}

int LeerSerial::unir(char a, char b){
    int n = 0;
    n = (int) ((a & 0x00FF)<<8) + (b & 0x00FF);
    return n;
}

void LeerSerial::deciferSmsg(char *data){
    int aux1, aux2;
    if(Rx tot <=999)
        Rx_bue++;
```

```
    MutexRefVars.lock();
       int eTX = RefVars.Estado_Tx;
    MutexRefVars.unlock();
    //qDebug()<<"eTX: "<<eTX;
    MutexEVars.lock();
        EVars.Est = (int) (data[9] & 0x00FF);
        EVars.Est_sig = (int) (data[10] & 0x00FF);
        if(eTX==1){
            aux1 = unir(data[1],data[2]);
            aux2 = unir(data[3],data[4]);
            //qDebug()<<"dat3: "<<((int) (data[3] & 0x00FF));
            //qDebug()<<"dat4: "<<((int) (data[4] & 0x00FF));
            if(aux1>20 && aux1<500) EVars.VABC[0] = aux1;
            if(aux2>20 && aux2<500) EVars.VABC[2] = aux2;
            if(aux1>=0 && aux1<=20) EVars.VABC[0] = 0;
            if(aux2>=0 && aux2<=20) EVars.VABC[2] = 0;
            aux1 = unir(data[5],data[6]);
            aux2 = unir(data[7],data[8]);
            if(aux1>200 && aux1<11000) EVars.PABC[0] = aux1;
            if(aux2>200 && aux2<11000) EVars.PABC[2] = aux2;
            if(aux1>=0 && aux1<=200) EVars.PABC[0] = 0;
            if(aux2>=0 && aux2<=200) EVars.PABC[2] = 0;
            EVars.VABC[1] = 0;
            EVars.PABC[1] = 0;
            switch(EVars.Est){
                case 1:
                    EVars.VABC[0] = 0;
                    EVars.VABC[1] = 0;
                    EVars.PABC[0] = 0;
                    EVars.PABC[1] = 0;
                    EVars.PABC[2] = 0;
                    break;
                case 2:
                    EVars.VABC[1] = 0;
                    EVars.PABC[0] = 0;
                    EVars.PABC[1] = 0;
                    EVars.PABC[2] = 0;
                    break;
                case 3:
                    EVars.VABC[0] = EVars.VABC[2];
                    EVars.VABC[1] = EVars.VABC[2];
                    EVars.PABC[1] = EVars.PABC[0]-EVars.PABC[2];
                    break;
                case 4:
                    EVars.VABC[1] = EVars.VABC[0];
                    EVars.PABC[1] = EVars.PABC[0];
                    break;
                case 5:
                    EVars.VABC[0] = 0;
                    EVars.PABC[0] = 0;
                    EVars.VABC[1] = EVars.VABC[2];
                    EVars.PABC[1] = EVars.PABC[2];
                    break;
                case 6:
                    EVars.VABC[0] = 0;
                    EVars.PABC[0] = 0;
                    EVars.VABC[1] = EVars.VABC[2];
                    EVars.PABC[1] = EVars.PABC[2];
                    break;
                default:
                    EVars.VABC[1] = 0;
                    EVars.PABC[1] = 0;

            }
        }
        if((int)(data[12] & 0x0001)) EVars.K[0]=1; //Kgen
        if((int)(data[12] & 0x0002)) EVars.K[1]=1; //Kred
        if((int)(data[12] & 0x0004)) EVars.K[2]=1; //Kcons

        //qDebug()<<"V g:"<<EVars.VABC[0]<<"c: "<<EVars.VABC[1]<<"r: "<<EVars.VABC[2];
    MutexEVars.unlock();
```

```
    MutexMVars.lock();
            MVars.val =(int) (data[11] & 0x00FF);
    MutexMVars.unlock();
}

void LeerSerial::deciferGRmsg(char *data){
    int aux1, aux2, aux3;
    MutexEVars.lock();
        aux1 = unir(data[1],data[2]);
        aux2 = unir(data[3],data[4]);
        aux3 = unir(data[5],data[6]);
        if(aux1>20 && aux1<500) EVars.VABC[0] = aux1;
        if(aux2>20 && aux2<500) EVars.VABC[1] = aux2;
        if(aux3>20 && aux3<500) EVars.VABC[2] = aux3;
        if(aux1>=0 && aux1<=20) EVars.VABC[0] = 0;
        if(aux2>=0 && aux2<=20) EVars.VABC[1] = 0;
        if(aux3>=0 && aux3<=20) EVars.VABC[2] = 0;
        aux1 = unir(data[7],data[8]);
        aux2 = unir(data[9],data[10]);
        aux3 = unir(data[11],data[12]);
        if(aux1>200 && aux1<11000) EVars.PABC[0] = aux1;
        if(aux2>200 && aux2<11000) EVars.PABC[1] = aux2;
        if(aux3>200 && aux3<11000) EVars.PABC[2] = aux3;
        if(aux1>=0 && aux1<=200) EVars.PABC[0] = 0;
        if(aux2>=0 && aux2<=200) EVars.PABC[1] = 0;
        if(aux3>=0 && aux3<=200) EVars.PABC[2] = 0;
        //qDebug()<<"GRV a:"<<EVars.VABC[0]<<"b: "<<EVars.VABC[1]<<"c: "<<EVars.VABC[2];
        qDebug()<<"GRV a:"<<((int) (data[2] & 0x00FF))<<"b: "<<((int) (data[4] &
0x00FF))<<"c: "<<((int) (data[6] & 0x00FF));
    MutexEVars.unlock();
}

void LeerSerial::sendmsg(){
    char bufTx[5];
    if(Tx==1){
        MutexRefVars.lock();
                bufTx[0] = 'E';
                bufTx[1] = (RefVars.Est sig & 0xFF00)>>8;
                bufTx[2] = RefVars.Est sig & 0x00FF;
                bufTx[3] = 'x';
                bufTx[4] = '\n';
        MutexRefVars.unlock();
    }
    else{
        MutexRefVars.lock();
                bufTx[0] = 'T';
                bufTx[1] = (RefVars.Estado_Tx & 0xFF00)>>8;
                bufTx[2] = RefVars.Estado_Tx & 0x00FF;
                bufTx[3] = 'x';
                bufTx[4] = '\n';
        MutexRefVars.unlock();
    }
    if(!WriteFile(hFile, bufTx, 5, &dwBytesRead, NULL)){
        //error occurred. Report to user.
        qDebug()<<"send error"<<"\n";
    }
}
```

B.7. Main.c

```
#include "uwindow.h"
#include "serial.h"
#include <QApplication>

#ifndef MYGLOBALS_H
#define MYGLOBALS_H
#include <globals.h>
#endif

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    uWindow w;
    initEVars(&EVars);
    initMVars(&MVars);
    initRefVars(&RefVars);
    Serial *serial = new Serial();
    QObject::connect(serial->lee,SIGNAL(sigACT()),&w,SLOT(Act()));
    QObject::connect(&w,SIGNAL(sigTransmit()),serial->lee,SLOT(SerialRead()));
    w.show();

    return a.exec();
}
```

B.8. Serial.c

```
#include "serial.h"

Serial::Serial(QObject *parent) :
    QObject(parent)
{
    timer = new QTimer();
    thread = new QThread();
    lee = new LeerSerial();
    lee->moveToThread(thread);
    QObject::connect(timer,SIGNAL(timeout()),this,SLOT(SerialTime()));
    QObject::connect(thread,SIGNAL(started()),lee,SLOT(SerialRead()));
    QObject::connect(lee,SIGNAL(SerialReadEnd()),thread,SLOT(quit()));
    if(SerialOpen())
        qDebug()<<"Serial Open";
}

bool Serial::SerialOpen(){
    hFile = CreateFile(TEXT("COM3"),GENERIC_READ |
GENERIC_WRITE,0,0,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,0);
    if(hFile == NULL)
    {
        qDebug()<<"error en crear el pueto\n";
        CloseHandle(hFile);
        return false;
    }

    dcbSerialParams.BaudRate=CBR 9600;
    dcbSerialParams.ByteSize=8;
    dcbSerialParams.StopBits=ONESTOPBIT;
    dcbSerialParams.Parity=NOPARITY;
```

```
    if(!SetCommState(hFile, &dcbSerialParams)){
       qDebug()<<"error en prams\n";
       CloseHandle(hFile);
       return false;
    }


    timeouts.ReadIntervalTimeout=50;
    timeouts.ReadTotalTimeoutConstant=50;
    timeouts.ReadTotalTimeoutMultiplier=10;
    timeouts.WriteTotalTimeoutConstant=50;
    timeouts.WriteTotalTimeoutMultiplier=10;
    if(!SetCommTimeouts(hFile, &timeouts)){
        qDebug()<<"error en timeout\n";
        CloseHandle(hFile);
        return false;
    }
    timer->start(290);
    return true;
}

void Serial::SerialClose(){
    CloseHandle(hFile);
}

void Serial::SerialTime(){
    thread->start();
}
```

B.9. Uwindow.c

```
#include "uwindow.h"
#include "ui_uwindow.h"

#define STYH "background-repeat: no-repeat; background-position: center center; font: 75
14pt Helvetica-Light; color: rgb(28, 54, 100);"
#define STYH2 "background-repeat: no-repeat; background-position: center center; font:
75 16pt Helvetica-Light; color: rgb(28, 54, 100);"

uWindow::uWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::uWindow)
{
    ui->setupUi(this);
    this->setFixedSize(796,483);
    ui->Wid->setCurrentIndex(1);
    ui->Wid->setStyleSheet("background-image: url(:/pes/Iconos/Fgcr2.png); "
                           "background-repeat: no-repeat;"
                           "background-position: center center");
    button = ui->pbGCR;
    button->setEnabled(false);
    buttonE = ui->pbDet;
    ActSys(1);
    MA = 0;
    ui->pb_MA_M->setEnabled(false);
    ui->pb_MA_M->setStyleSheet("background-image: url(:/pes/Iconos/pbMA_MA2.png);"
STYH2);
    MutexRefVars.lock();
    RefVars.Estado_Tx = 1;
    MutexRefVars.unlock();
    Tx = 2;
    emit sigTransmit();
}
```

```
uWindow::~uWindow()
{
    delete ui;
}

void uWindow::on_pbGCR_clicked()
{
    button->setEnabled(true);
    button = ui->pbGCR;
    button->setEnabled(false);
    ui->Wid->setCurrentIndex(1);
    ui->Wid->setStyleSheet("background-image: url(:/pes/Iconos/Fgcr2.png); "
                           "background-repeat: no-repeat;"
                           "background-position: center center");

    ui->LabGCR_VG->setStyleSheet("background: rgb(203, 201, 229);"
                             "font: 25 14pt Helvetica-Light;"
                             "color: rgb(28, 54, 100);");
    ui->LabGCR_VC->setStyleSheet("background: rgb(203, 201, 229);"
                             "font: 25 14pt Helvetica-Light;"
                             "color: rgb(28, 54, 100);");
    ui->LabGCR_VR->setStyleSheet("background: rgb(203, 201, 229);"
                             "font: 25 14pt Helvetica-Light;"
                             "color: rgb(28, 54, 100);");
    ui->LabGCR_PG->setStyleSheet("background: rgb(203, 201, 229);"
                             "font: 25 14pt Helvetica-Light;"
                             "color: rgb(28, 54, 100);");
    ui->LabGCR PC->setStyleSheet("background: rgb(203, 201, 229);"
                             "font: 25 14pt Helvetica-Light;"
                             "color: rgb(28, 54, 100);");
    ui->LabGCR_PR->setStyleSheet("background: rgb(203, 201, 229);"
                             "font: 25 14pt Helvetica-Light;"
                             "color: rgb(28, 54, 100);");
    MutexRefVars.lock();
    RefVars.Estado_Tx = 1;
    MutexRefVars.unlock();
    Tx = 2;
    emit sigTransmit();
}

void uWindow::on_pbMA_clicked()
{
    button->setEnabled(true);
    button = ui->pbMA;
    button->setEnabled(false);
    ui->Wid->setCurrentIndex(0);
    ui->Wid->setStyleSheet("background: white");
    MutexRefVars.lock();
    RefVars.Estado_Tx = 1;
    MutexRefVars.unlock();
    Tx = 2;
    emit sigTransmit();
}


void uWindow::on_pbG_clicked()
{
    button->setEnabled(true);
    button = ui->pbG;
    button->setEnabled(false);
    ui->Wid->setCurrentIndex(2);
    ui->Wid->setStyleSheet("background-image: url(:/pes/Iconos/Fg2.png); "
                           "background-repeat: no-repeat;"
                           "background-position: center center");
    ui->LabVA->setStyleSheet("background: rgb(203, 201, 229);"
                             "font: 25 14pt Helvetica-Light;"
                             "color: rgb(28, 54, 100);");
    ui->LabVB->setStyleSheet("background: rgb(203, 201, 229);"
                             "font: 25 14pt Helvetica-Light;"
                             "color: rgb(28, 54, 100);");
    ui->LabVC->setStyleSheet("background: rgb(203, 201, 229);"
```

```
                            "font: 25 14pt Helvetica-Light;"
                            "color: rgb(28, 54, 100);");
    ui->LabPA->setStyleSheet("background: rgb(203, 201, 229);"
                            "font: 25 14pt Helvetica-Light;"
                            "color: rgb(28, 54, 100);");
    ui->LabPB->setStyleSheet("background: rgb(203, 201, 229);"
                            "font: 25 14pt Helvetica-Light;"
                            "color: rgb(28, 54, 100);");
    ui->LabPC->setStyleSheet("background: rgb(203, 201, 229);"
                            "font: 25 14pt Helvetica-Light;"
                            "color: rgb(28, 54, 100);");
    MutexRefVars.lock();
    RefVars.Estado_Tx = 2;
    MutexRefVars.unlock();
    Tx = 2;
    emit sigTransmit();
}

void uWindow::on_pbR_clicked()
{
    button->setEnabled(true);
    button = ui->pbR;
    button->setEnabled(false);
    ui->Wid->setCurrentIndex(2);
    ui->Wid->setStyleSheet("background-image: url(:/pes/Iconos/Fr2.png); "
                           "background-repeat: no-repeat;"
                           "background-position: center center");
    ui->LabVA->setStyleSheet("background: rgb(203, 201, 229);"
                            "font: 25 14pt Helvetica-Light;"
                            "color: rgb(28, 54, 100);");
    ui->LabVB->setStyleSheet("background: rgb(203, 201, 229);"
                            "font: 25 14pt Helvetica-Light;"
                            "color: rgb(28, 54, 100);");
    ui->LabVC->setStyleSheet("background: rgb(203, 201, 229);"
                            "font: 25 14pt Helvetica-Light;"
                            "color: rgb(28, 54, 100);");
    ui->LabPA->setStyleSheet("background: rgb(203, 201, 229);"
                            "font: 25 14pt Helvetica-Light;"
                            "color: rgb(28, 54, 100);");
    ui->LabPB->setStyleSheet("background: rgb(203, 201, 229);"
                            "font: 25 14pt Helvetica-Light;"
                            "color: rgb(28, 54, 100);");
    ui->LabPC->setStyleSheet("background: rgb(203, 201, 229);"
                            "font: 25 14pt Helvetica-Light;"
                            "color: rgb(28, 54, 100);");
    MutexRefVars.lock();
    RefVars.Estado_Tx = 3;
    MutexRefVars.unlock();
    Tx = 2;
    emit sigTransmit();
}

void uWindow::on_pbAL_clicked()
{
    button->setEnabled(true);
    button = ui->pbAL;
    button->setEnabled(false);
    ui->Wid->setCurrentIndex(3);
    ui->Wid->setStyleSheet("background-image: url(:/pes/Iconos/alarmas-cabecera.png); "
                           "background-repeat: no-repeat;"
                           "background-position: top center");
    MutexRefVars.lock();
    RefVars.Estado Tx = 1;
    MutexRefVars.unlock();
    Tx = 2;
    emit sigTransmit();
}

void uWindow::on pbHR clicked()
{
    button->setEnabled(true);
```

```
    button = ui->pbHR;
    button->setEnabled(false);
    ui->Wid->setCurrentIndex(4);
    ui->Wid->setStyleSheet("background: white");
    MutexRefVars.lock();
    RefVars.Estado_Tx = 1;
    MutexRefVars.unlock();
    Tx = 2;
    emit sigTransmit();
}

void uWindow::on_pbRG_clicked()
{
    button->setEnabled(true);
    button = ui->pbRG;
    button->setEnabled(false);
    ui->Wid->setCurrentIndex(5);
    ui->Wid->setStyleSheet("background-image: url(:/pes/Iconos/registro-cabecera.png);"
                           "background-repeat: no-repeat;"
                           "background-position: top center;");
    ui->LabRG D1->setStyleSheet("background: rgb(203, 201, 229);"
                                "font: 25 12pt Helvetica-Light;"
                                "color: rgb(28, 54, 100);");
    ui->LabRG_D2->setStyleSheet("background: rgb(203, 201, 229);"
                                "font: 25 12pt Helvetica-Light;"
                                "color: rgb(28, 54, 100);");
    ui->LabRG D3->setStyleSheet("background: rgb(203, 201, 229);"
                                "font: 25 12pt Helvetica-Light;"
                                "color: rgb(28, 54, 100);");
    MutexRefVars.lock();
    RefVars.Estado_Tx = 1;
    MutexRefVars.unlock();
    Tx = 2;
    emit sigTransmit();
}

void uWindow::on_pbAY_clicked()
{
    button->setEnabled(true);
    button = ui->pbAY;
    button->setEnabled(false);
    ui->Wid->setCurrentIndex(6);
    ui->Wid->setStyleSheet("background: white");
    MutexRefVars.lock();
    RefVars.Estado_Tx = 1;
    MutexRefVars.unlock();
    Tx = 2;
    emit sigTransmit();
}

void uWindow::on pb MA M clicked()
{
    ui->pb_MA_M->setEnabled(false);
    ui->pb_MA_A->setEnabled(true);
    ui->pb MA M->setStyleSheet("background-image: url(:/pes/Iconos/pbMA MA2.png);"
STYH2);
    ui->pb MA A->setStyleSheet("background-image: url(:/pes/Iconos/pbMA MA.png);"
STYH2);
    ui->pbDet->setEnabled(true);
    ui->pbEsp->setEnabled(true);
    ui->pbIsl->setEnabled(true);
    ui->pbSic->setEnabled(true);
    ui->pbRed->setEnabled(true);
    ui->pbMan->setEnabled(true);
    ui->pbDet->setStyleSheet("background-image: url(:/pes/Iconos/pb1.png);" STYH);
    ui->pbEsp->setStyleSheet("background-image: url(:/pes/Iconos/pb1.png);" STYH);
    ui->pbIsl->setStyleSheet("background-image: url(:/pes/Iconos/pb1.png);" STYH);
    ui->pbSic->setStyleSheet("background-image: url(:/pes/Iconos/pb1.png);" STYH);
    ui->pbRed->setStyleSheet("background-image: url(:/pes/Iconos/pb1.png);" STYH);
    ui->pbMan->setStyleSheet("background-image: url(:/pes/Iconos/pb1.png);" STYH);
    ActSys(1);
```

```
    MA = 0;
}

void uWindow::on_pb_MA_A_clicked()
{
    ui->pb_MA_M->setEnabled(true);
    ui->pb_MA_A->setEnabled(false);
    ui->pb_MA_M->setStyleSheet("background-image: url(:/pes/Iconos/pbMA_MA.png);"
STYH2);
    ui->pb_MA_A->setStyleSheet("background-image: url(:/pes/Iconos/pbMA_MA2.png);"
STYH2);
    ui->pbDet->setEnabled(false);
    ui->pbEsp->setEnabled(false);
    ui->pbIsl->setEnabled(false);
    ui->pbSic->setEnabled(false);
    ui->pbRed->setEnabled(false);
    ui->pbMan->setEnabled(false);
    ui->pbDet->setStyleSheet("background-image: url(:/pes/Iconos/pb2.png);" STYH);
    ui->pbEsp->setStyleSheet("background-image: url(:/pes/Iconos/pb2.png);" STYH);
    ui->pbIsl->setStyleSheet("background-image: url(:/pes/Iconos/pb2.png);" STYH);
    ui->pbSic->setStyleSheet("background-image: url(:/pes/Iconos/pb2.png);" STYH);
    ui->pbRed->setStyleSheet("background-image: url(:/pes/Iconos/pb2.png);" STYH);
    ui->pbMan->setStyleSheet("background-image: url(:/pes/Iconos/pb2.png);" STYH);
    MA = 1;
}

void uWindow::ActSys(int e){
    switch(e){
        case 1:
            ui->LabK->setPixmap(QPixmap(":/Fondo/Iconos/Edetenido.png"));
            ui->LabEst->setText("DETENIDO");
            if(!MA){
                buttonE->setEnabled(true);
                buttonE->setStyleSheet("background-image: url(:/pes/Iconos/pb1.png);"
STYH);
            }
            ui->pbDet->setEnabled(false);
            ui->pbDet->setStyleSheet("background-image: url(:/pes/Iconos/pb2.png);"
STYH);
            buttonE = ui->pbDet;

            break;
        case 2:
            ui->LabK->setPixmap(QPixmap(":/Fondo/Iconos/Eespera.png"));
            ui->LabEst->setText("EN ESPERA ...");
            if(!MA){
                buttonE->setEnabled(true);
                buttonE->setStyleSheet("background-image: url(:/pes/Iconos/pb1.png);"
STYH);
            }
            ui->pbEsp->setEnabled(false);
            ui->pbEsp->setStyleSheet("background-image: url(:/pes/Iconos/pb2.png);"
STYH);
            buttonE = ui->pbEsp;
            break;
        case 3:
            ui->LabK->setPixmap(QPixmap(":/Fondo/Iconos/Esincronizado.png"));
            ui->LabEst->setText("SINCRONIZADO");
            if(!MA){
                buttonE->setEnabled(true);
                buttonE->setStyleSheet("background-image: url(:/pes/Iconos/pb1.png);"
STYH);
            }
            ui->pbSic->setEnabled(false);
            ui->pbSic->setStyleSheet("background-image: url(:/pes/Iconos/pb2.png);"
STYH);
            buttonE = ui->pbSic;
            break;
        case 4:
            ui->LabK->setPixmap(QPixmap(":/Fondo/Iconos/Eisla.png"));
            ui->LabEst->setText("ISLA");
```

```
            if(!MA){
                buttonE->setEnabled(true);
                buttonE->setStyleSheet("background-image: url(:/pes/Iconos/pb1.png);"
STYH);
            }
            ui->pbIsl->setEnabled(false);
            ui->pbIsl->setStyleSheet("background-image: url(:/pes/Iconos/pb2.png);"
STYH);
            buttonE = ui->pbIsl;
            break;
        case 5:
            ui->LabK->setPixmap(QPixmap(":/Fondo/Iconos/Ered.png"));
            ui->LabEst->setText("RED");
            if(!MA){
                buttonE->setEnabled(true);
                buttonE->setStyleSheet("background-image: url(:/pes/Iconos/pb1.png);"
STYH);
            }
            ui->pbRed->setEnabled(false);
            ui->pbRed->setStyleSheet("background-image: url(:/pes/Iconos/pb2.png);"
STYH);
            buttonE = ui->pbRed;
            break;
        case 6:
            ui->LabK->setPixmap(QPixmap(":/Fondo/Iconos/Emantension.png"));
            ui->LabEst->setText("MANTENCIÓN");
            if(!MA){
                buttonE->setEnabled(true);
                buttonE->setStyleSheet("background-image: url(:/pes/Iconos/pb1.png);"
STYH);
            }
            ui->pbMan->setEnabled(false);
            ui->pbMan->setStyleSheet("background-image: url(:/pes/Iconos/pb2.png);"
STYH);
            buttonE = ui->pbMan;
            break;
        default:
            ui->LabEst->setText("ERROR");
    }
}

void uWindow::on_pbDet_clicked()
{
    MutexRefVars.lock();
    RefVars.Est_sig = 1;
    MutexRefVars.unlock();
    Tx = 1;
    emit sigTransmit();
}

void uWindow::on pbEsp clicked()
{
    MutexRefVars.lock();
    RefVars.Est_sig = 2;
    MutexRefVars.unlock();
    Tx = 1;
    emit sigTransmit();
}

void uWindow::on_pbSic_clicked()
{
    MutexRefVars.lock();
    RefVars.Est sig = 3;
    MutexRefVars.unlock();
    Tx = 1;
    emit sigTransmit();
}

void uWindow::on pbIsl clicked()
{
    MutexRefVars.lock();
```

```
    RefVars.Est_sig = 4;
    MutexRefVars.unlock();
    Tx = 1;
    emit sigTransmit();
}

void uWindow::on_pbRed_clicked()
{
    MutexRefVars.lock();
    RefVars.Est_sig = 5;
    MutexRefVars.unlock();
    Tx = 1;
    emit sigTransmit();
}

void uWindow::on_pbMan_clicked()
{
    MutexRefVars.lock();
    RefVars.Est_sig = 6;
    MutexRefVars.unlock();
    Tx = 1;
    emit sigTransmit();
}

void uWindow::Act(){
    switch(ui->Wid->currentIndex()){
        case 0:
            ActMA();
            break;
        case 1:
            ActGCR();
            break;
        case 2:
        case 3:
            ActGR();
            break;
        default:
            break;
    }
}

void uWindow::ActMA(){
    MutexEVars.lock();
        ActSys(EVars.Est);
    MutexEVars.unlock();
}
void uWindow::ActGCR(){
    MutexEVars.lock();
        ActSys(EVars.Est);
        ui->LabGCR_VG->setText("V: " + QString::number(EVars.VABC[0]) + " [V]");
        ui->LabGCR_VC->setText("V: " + QString::number(EVars.VABC[1]) + " [V]");
        ui->LabGCR_VR->setText("V: " + QString::number(EVars.VABC[2]) + " [V]");
        ui->LabGCR_PG->setText("P: " + QString::number(EVars.PABC[0]*0.001) + " [kW]");
        ui->LabGCR_PC->setText("P: " + QString::number(EVars.PABC[1]*0.001) + " [kW]");
        ui->LabGCR_PR->setText("P: " + QString::number(EVars.PABC[2]*0.001) + " [kW]");
    MutexEVars.unlock();
}
void uWindow::ActGR(){
    MutexEVars.lock();
        ActSys(EVars.Est);
        ui->LabVA->setText("V: " + QString::number(EVars.VABC[0]) + " [V]");
        ui->LabVB->setText("V: " + QString::number(EVars.VABC[1]) + " [V]");
        ui->LabVC->setText("V: " + QString::number(EVars.VABC[2]) + " [V]");
        ui->LabPA->setText("P: " + QString::number(EVars.PABC[0]*0.001) + " [kW]");
        ui->LabPB->setText("P: " + QString::number(EVars.PABC[1]*0.001) + " [kW]");
        ui->LabPC->setText("P: " + QString::number(EVars.PABC[2]*0.001) + " [kW]");
    MutexEVars.unlock();
}
```

C. Programa de prueba de comunicación

C.1. Lab9.c

```
//
//      Lab9 4: TMS320F28335
//      (c) Frank Bormann
//
//###########################################################################
//
// FILE:       Lab9_4.c
//
// TITLE:      DSP28 SCI - Communication to PC - Terminal
//                     SCI-Setup: 9600 Baud, 8 Bit , ODD Parity , 1 Stopbit
//                     SCI - TX and RX - Interrupt are used in this Lab
//                     SCI - TX and RX - FIFO are used in this Lab
//                     DSP waits for "Texas" and answers with "Instruments"
//                     Watchdog active , serviced solely in main-loop
//###########################################################################
//  Ver | dd mmm yyyy | Who  | Description of changes
// =====|=============|======|===============================================
//  3.0 | 08 Jul 2009 | F.B. | adapted for ControlCard28335 @ 20MHz
//  3.1 | 15 Nov 2009 | F.B  | Lab9 4 for F28335 @30MHz and PE revision 5
//###########################################################################
#include "DSP2833x Device.h"
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

// External Function prototypes
extern void InitSysCtrl(void);
extern void InitPieCtrl(void);
extern void InitPieVectTable(void);
extern void InitCpuTimers(void);
extern void ConfigCpuTimer(struct CPUTIMER VARS *, float, float);

// Prototype statements for functions found within this file.
void Gpio_select(void);
void SCIA_init(void);
interrupt void SCIA_TX_isr(void);     // SCI-A Transmit Interrupt Service
interrupt void SCIA RX isr(void);     // SCI-A Receive  Interrupt Service
interrupt void cpu timer0 isr(void); // Prototype for Timer 0 Interrupt Service Routine

// Global Variables
char message[]={" Instruments! \n\r"};
char message2[]={" Timer isr!!! \n\r"};
char buffer[16];
int BT=1;
int isr = 0;
//###########################################################################
//                                          main code

//###########################################################################
void main(void)
{
       InitSysCtrl(); // Basic Core Initialization
                                  // SYSCLK=150MHz, HISPCLK=75MHz, LSPCLK=37.5MHz
       EALLOW;
       SysCtrlRegs.WDCR= 0x00AF;             // Re-enable the watchdog
       EDIS;                 // 0x00E8  to disable the Watchdog , Prescaler = 1
                                  // 0x00AF  to NOT disable the Watchdog, Prescaler =
64

       Gpio select(); // GPIO9, GPIO11, GPIO34 and GPIO49 as output
                                  // to 4 LEDs at Peripheral Explorer
```

```
        InitPieCtrl();        // default status of PIE; in DSP2833x_PieCtrl.c

        InitPieVectTable(); // init PIE vector table; in DSP2833x_PieVect.c

        InitCpuTimers();
        ConfigCpuTimer(&CpuTimer0, 150, 1000000);   // DSP2833x_CpuTimers.c

        // re-map PIE - entry for SCI-A-TX and SCI-A-RX
        EALLOW;
        PieVectTable.SCITXINTA = &SCIA_TX_isr;
        PieVectTable.SCIRXINTA = &SCIA_RX_isr;
        PieVectTable.TINT0 = &cpu_timer0_isr;
        EDIS;

        SCIA_init();  // Initalize SCI

    // Enable SCI-A TX Interrupt Group9 interupt 2
        PieCtrlRegs.PIEIER9.bit.INTx2 = 1;
        // Enable SCI-A RX Interrupt Group9 interupt 1
        PieCtrlRegs.PIEIER9.bit.INTx1 = 1;
        //timer interrupt enable
        PieCtrlRegs.PIEIER1.bit.INTx7 = 1;


        // Enable INT9 for SCIA-TX and SCIA-RX:
    IER = 0x100;
    IER |= 0x001; //timer 0
        // Enable global Interrupts and higher priority real-time debug events:
        EINT;   // Enable Global interrupt INTM
        ERTM;   // Enable Global realtime interrupt DBGM
        CpuTimer0Regs.TCR.bit.TSS = 0;
        while(1)
        {
               EALLOW;
               SysCtrlRegs.WDKEY = 0x55;                    // Service watchdog #1
               SysCtrlRegs.WDKEY = 0xAA;                    // Service watchdog #2
               EDIS;
        }
}

void Gpio_select(void)
{
        EALLOW;
        GpioCtrlRegs.GPAMUX1.all = 0;        // GPIO15 ... GPIO0 = General Puropse I/O
        GpioCtrlRegs.GPAMUX2.all = 0;        // GPIO31 ... GPIO16 = General Purpose I/O

        GpioCtrlRegs.GPAMUX2.bit.GPIO28 = 1; // SCIRXDA
        GpioCtrlRegs.GPAMUX2.bit.GPIO29 = 1; // SCITXDA

        GpioCtrlRegs.GPBMUX1.all = 0;        // GPIO47 ... GPIO32 = General Purpose I/O
        GpioCtrlRegs.GPBMUX2.all = 0;        // GPIO63 ... GPIO48 = General Purpose I/O
        GpioCtrlRegs.GPCMUX1.all = 0;        // GPIO79 ... GPIO64 = General Purpose I/O
        GpioCtrlRegs.GPCMUX2.all = 0;        // GPIO87 ... GPIO80 = General Purpose I/O

        GpioCtrlRegs.GPADIR.all = 0;
        GpioCtrlRegs.GPADIR.bit.GPIO9 = 1;   // peripheral explorer: LED LD1 at GPIO9
        GpioCtrlRegs.GPADIR.bit.GPIO11 = 1;  // peripheral explorer: LED LD2 at GPIO11

        GpioCtrlRegs.GPBDIR.all = 0;         // GPIO63-32 as inputs
        GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1;  // peripheral explorer: LED LD3 at GPIO34
        GpioCtrlRegs.GPBDIR.bit.GPIO49 = 1; // peripheral explorer: LED LD4 at GPIO49

        GpioCtrlRegs.GPCDIR.all = 0;         // GPIO87-64 as inputs
        EDIS;
}

void SCIA init()
{
        SciaRegs.SCICCR.all =0x0007;  // 1 stop bit,  No loopback
                                      // no parity,8 char bits,
```

```
                                    // async mode, idle-line protocol
       SciaRegs.SCICTL1.all =0x0003; // enable TX, RX, internal SCICLK,
                                    // Disable RX ERR, SLEEP, TXWAKE

       // SYSCLOCKOUT = 150MHz; LSPCLK = 1/4 = 37.5 MHz
       // BRR = (LSPCLK / (9600 x 8)) -1
       // BRR = 487  gives 9605 Baud
       SciaRegs.SCIHBAUD    = 487 >> 8;            // Highbyte
       SciaRegs.SCILBAUD    = 487 & 0x00FF; // Lowbyte

       SciaRegs.SCICTL2.bit.TXINTENA = 1;          // enable SCI-A Tx-ISR
       SciaRegs.SCICTL2.bit.RXBKINTENA = 1;        // enable SCI_A Rx-ISR

       SciaRegs.SCIFFTX.all = 0xC060;       // bit 15 = 1 : relinquish from Reset
                                            // bit 14 = 1 : Enable FIFO
                                           // bit 6 = 1 :  CLR TXFFINT-Flag
                                            // bit 5 = 1 :  enable TX FIFO match
                                            // bit 4-0 :  TX-ISR, if TX FIFO is 0(empty)
       SciaRegs.SCIFFCT.all = 0x0000;       // Set FIFO transfer delay to 0

       SciaRegs.SCIFFRX.all = 0xE065;       // Rx interrupt level = 5

       SciaRegs.SCICTL1.all = 0x0023;       // Relinquish SCI from Reset
}

// SCI-A Transmit Interrupt Service
interrupt void SCIA TX isr(void)
{
       unsigned int i;
       // copy 16 character into SCI-A TX buffer
       if (BT==1){
              for(i=0;i<16;i++) SciaRegs.SCITXBUF= message[i];
       }
       else{
              for(i=0;i<16;i++) SciaRegs.SCITXBUF= message2[i];
       }
       BT=0;
       // Acknowledge this interrupt to receive more interrupts from group 9
    PieCtrlRegs.PIEACK.all = PIEACK GROUP9;
}

// SCI-A Receive Interrupt Service
interrupt void SCIA_RX_isr(void)
{
       int i;
       for (i=0;i<16;i++) buffer[i]= SciaRegs.SCIRXBUF.bit.RXDT;

       if (strncmp(buffer, "Texas", 5) == 0)
       {
              BT=1;
              SciaRegs.SCIFFTX.bit.TXFIFOXRESET =1;  // enable TXFIFO
              SciaRegs.SCIFFTX.bit.TXFFINTCLR = 1 ;  // force TX-ISR
       }
       SciaRegs.SCIFFRX.bit.RXFIFORESET = 0;       // reset RX-FIFO pointer
       SciaRegs.SCIFFRX.bit.RXFIFORESET = 1;       // enable RX-operation
       SciaRegs.SCIFFRX.bit.RXFFINTCLR = 1;    // clear RX-FIFO INT Flag
       isr++;
       PieCtrlRegs.PIEACK.all = PIEACK GROUP9;
}


interrupt void cpu_timer0_isr(void)
{
    CpuTimer0.InterruptCount++;             // increment time counter
       GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1; // toggle LED at GPIO34
       // Service the watchdog every Timer 0 interrupt
       EALLOW;
       SysCtrlRegs.WDKEY = 0x55;            // Service watchdog #1
       EDIS;

       SciaRegs.SCIFFTX.bit.TXFIFOXRESET =1;  // enable TXFIFO
```

```
        SciaRegs.SCIFFTX.bit.TXFFINTCLR = 1 ;  // force TX-ISR
        // Acknowledge this interrupt to receive more interrupts from group 1
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}
//===========================================================================
// End of SourceCode.
//===========================================================================
```