Universidad Nacional Pedro Ruiz Gallo

Facultad de Ciencias Físicas y Matemáticas

Escuela Profesional de Ingeniería Electrónica

TESIS:

DISEÑO Y CONSTRUCCIÓN DE UNA MINI-PLANTA DE CONTROL DE NIVEL BASADO EN MICROCONTROLADORES DE 32 BITS COMO GUIA PARA EL CAMPO DE ELECTRONICA DIGITAL PARA LA ESCUELA DE INGENIERIA ELECTRONICA UNPRG

Tesis presentada por:

Sergio Bacheli Acuña Cieza

Ronal Roiser Fernandez Sempertegui

Asesor:

Ing. Carlos Leonardo Oblitas Vera

Lambayeque - Lambayeque

Septiembre 2016

Universidad Nacional Pedro Ruiz Gallo

Facultad de Ciencias Físicas y Matemáticas

Escuela Profesional de Ingeniería Electrónica

Tesis presentada para obtener el grado de

Ingeniero Electrónico

DISEÑO Y CONSTRUCCIÓN DE UNA MINI-PLANTA DE CONTROL DE NIVEL BASADO EN MICROCONTROLADORES DE 32 BITS COMO GUIA PARA EL CAMPO DE ELECTRONICA DIGITAL PARA LA ESCUELA DE INGENIERIA ELECTRONICA UNPRG

Por

Sergio Bacheli Acuña Cieza

Ronal Roiser Fernandez Sempertegui

Lambayeque - Lambayeque

Septiembre 2016

DISEÑO Y CONSTRUCCIÓN DE UNA MINI-PLANTA DE CONTROL DE NIVEL BASADO EN MICROCONTROLADORES DE 32 BITS COMO GUIA PARA EL CAMPO DE ELECTRONICA DIGITAL PARA LA ESCUELA DE INGENIERIA ELECTRONICA UNPRG

Sergio Bacheli Acuña Cieza

Ronal Roiser Fernandez Sempertegui

Lambayeque - Lambayeque

Septiembre 2016

Tesis presentada por:

Sergio Bacheli Acuña Cieza

Ronal Roiser Fernandez Sempertegui

Como requisito para obtener el título de

Ingeniero en Electrónica.

Aceptada por la Escuela Profesional de Ingeniería Electrónica

Ing. Manuel Javier Ramírez Castro Presidente	Ing. Hugo Chiclayo Padilla Secretario
Ing. Lucía Isabel Chamán Cabrera Vocal	Ing. Carlos Leonardo Oblitas Vera Asesor
Sergio Bacheli Acuña Cieza Autor	Ronal Roiser Fernández Sempertegui Autor

Lambayeque - Lambayeque

Septiembre 2016

Dedicatoria

A mis padres por su apoyo, consejos, comprensión, amor, ayuda en los momentos difíciles, y por ayudarme con los recursos necesarios para estudiar. Me han dado todo lo que soy como persona, mis valores, mis principios, mi carácter, mi empeño, mi perseverancia, mi coraje para conseguir mis objetivos.

Bacheli Acuña

Dedicatoria

Con todo mi cariño y mi amor a esas personas importantes en mi vida, que siempre estuvieron listas para brindarme toda su ayuda, ahora me toca regresar un poquito de todo lo inmenso que me han otorgado. Con todo mi cariño está tesis se las dedico a ustedes: Papa y Mama.

Roiser Fernandez

Agradecimiento

El presente trabajo de tesis en primer lugar nos gustaría agradecer a ti Dios por bendecirnos para llegar hasta donde hemos llegado, porque hiciste realidad este sueño anhelado.

A nuestros padres, por todo el apoyo brindado a lo largo de nuestras vidas. Por darnos la oportunidad de estudiar esta carrera profesional, y ser nuestros ejemplos de vida, y por promovernos el desarrollo y la unión en nuestras familias.

A la UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO, por darnos la oportunidad de estudiar y ser profesionales.

A nuestros maestros, que en este andar por la vida, influyeron con sus lecciones y experiencias en formarnos como personas de bien y preparadas para los retos que pone la vida.

Son muchas las personas que han formado parte de nuestra vida profesional a las que nos encantaría agradecerles su amistad, consejos, apoyo, ánimo y compañía en los momentos más difíciles de nuestras vidas. Algunas están aquí con nosotros y otras en nuestros recuerdos y en nuestros corazones, sin importar en donde estén queremos darles las gracias por formar parte de nosotros, por todo lo que nos han brindado y por todas sus bendiciones.

Los Autores

Introducción

Para los procesos de producción que requieren del suministro de algún líquido como materia prima, dispensados desde altura, es imprescindible la constante revisión del nivel del líquido en el depósito, como también del caudal que ingresa y sale del mismo, para lograr controlar un proceso es importante tratarlo como un sistema continuo en el tiempo, en el cual cada una de sus partes, cumple una función y se interrelaciona con las demás.

Todo sistema continuo en el tiempo puede ser representado a través de una función de transferencia, la cual es una expresión matemática del modelo del sistema, formada por el cociente de dos polinomios, expresados en transformada de Laplace.

Dentro de los dispositivos digitales más resaltantes tenemos a los microcontroladores de 8 bits, a los cuales se les puede programar con unas cuantas líneas de código, pero no todos son de la rama industrial y mucho menos son capaces de almacenar una ecuación de un sistema de control. Para tal inconveniente también existen los microcontroladores de 32 bits con muchas más cualidades industriales (como mayor velocidad, mayor performance, mayor capacidad de memoria, más módulos de comunicación interna, etc.) lo cuales los pueden convertir en el guerrero de batalla para cualquier sistema de control.

Actualmente en la escuela de Ingeniería Electrónica se cuenta con una mini-planta de control de nivel de líquido gobernado por un dispositivo lógico programable industrial que es un PLC de la marca Schneider Electric más aun no se ha logrado evolucionar en el campo de electrónica digital y poder desarrollar proyectos y dispositivos de igual potencia como los de cualquier marca reconocida en el marcado actual.

Resumen

El presente proyecto tiene por objetivo el diseñar y construir una Mini Planta de Control de Nivel basado en los Microcontroladores de 32 bits como Guía para el campo de la Electrónica Digital, que complementen la temática desarrollada en los cursos impartidos por la Escuela de Profesional de Ingeniería Electrónica de la Universidad Nacional Pedro Ruiz Gallo.

El CPU del proceso es la Placa de Evaluación Stm32-V3, que se encargará de procesar las señales obtenidas del sensor Ultrasónico (su funcionamiento es simple: se encarga de emitir un pulso de ultrasonido que rebota sobre el agua y calcula la distancia midiendo el tiempo que transcurre entre la emisión del sonido y la percepción del eco).

La Placa de Evaluación compara la variable del proceso, que es la altura del nivel de agua con el valor deseado y envía su respuesta al variador de velocidad (Placa de Evaluación) y a la válvula. Este a su vez controla a la electroválvula, aumentando y/o disminuyendo el caudal de entrada al tanque.

En este sistema también podemos agregar perturbaciones externas a través de las válvulas manuales. Pero la característica principal de este tipo de control es que le hace inmune a las perturbaciones, obteniendo la altura deseada.

Abstract

The objective of this project is to design and build a Mini Level Control Plant based on 32-bit microcontrollers as the guide for the field of Digital Electronics, which complements the thematic developed in the courses taught by the School of Professional Electronic Engineering of the National University Pedro Ruiz Gallo.

The CPU of the process is the test of Stm32-V3, which is in charge of processing the signals obtained from the Ultrasonic sensor (its operation is simple: it is responsible for emitting an ultrasound pulse that bounces over the water and calculates the distance by measuring the time that Between the emission of sound and the perception of the echo).

The Evaluation Plate compares the process variable, which is the height of the water level to the desired value and sends its response to the variable speed controller (Evaluation Plate) and to the valve. This in turn controls the solenoid valve, increasing and / or decreasing the inflow to the tank.

In this system external disturbances can also be added through the manual valves. But the main characteristic of this type of control is that it makes immune to the disturbances, getting the desired height.

Índice general

Índice de figuras

CAPÍTULO 1

Aspectos Generales del Proyecto

1.1 ASPECTOS GENERALES

1.1.1. Título del Proyecto

DISEÑO Y CONSTRUCCIÓN DE UNA MINI-PLANTA DE CONTROL DE NIVEL BASADO EN MICROCONTROLADORES DE 32 BITS COMO GUIA PARA EL CAMPO DE ELECTRONICA DIGITAL PARA LA ESCUELA DE INGENIERIA ELECTRONICA UNPRG

1.1.2. Definición del Problema

¿Cómo diseñar y construir una mini-planta de control de nivel basado en microcontroladores de 32 bits como guía para el campo de electrónica digital para la escuela de ingeniería electrónica UNPRG?

1.1.3. Formulación de la Hipótesis

Se puede diseñar y construir una mini-planta de control de nivel basado en microcontroladores de 32 bits y que servirá como guía para el campo de electrónica digital para la escuela de ingeniería electrónica UNPRG.

1.1.4. Objetivos del Proyecto

1.1.4.1. Objetivo General

Diseñar y construir una mini-planta de control de nivel basado en microcontroladores de 32 bits.

1.1.4.2. Objetivos Específicos

- Enmarcar todos los conceptos y criterios teóricos necesarios para el diseño de una mini-planta de control de nivel.
- Plantear el diseño estructural de la mini-planta basada en un modelo conocido.
- Seleccionar sensores y actuadores a utilizar para su óptimo funcionamiento
- Plantear el diseño del sistema de control basado en un Micro-controlador de 32 bits
- Incentivar la participación de los alumnos y docentes con el uso de micro-controladores de gama industrial.
- Realizar pruebas del funcionamiento

1.1.5. Justificación e Importancia del Proyecto

Esta investigación nace para los Alumnos de la Escuela Profesional de Ingeniería Electrónica de la Universidad Nacional Pedro Ruiz Gallos, y es importante, porque:

- Incrementa la motivación y el interés de alumnos a desarrollar diseños tecnológicos de gama industrial.
- Ayuda a comprender y poder practicar las diferentes teorías de control digital impartidas en clase y así mismo ayuda a enlazar las ramas de control industrial con la programación de dispositivos digitales.
- Servirá de guía para los futuros alumnos de la escuela de ingeniería electrónica y a ser incentivo para que ellos pueda mejorar el diseño propuesto e incluso crear sus propios proyectos electrónicos.

1.1.6. Antecedentes de la Escuela Profesional de Ingeniería Electrónica

La Escuela Profesional de Ingeniería Electrónica, se crea por Resolución N° 007-98-AU-R y se adscribe a la Facultad de Ciencias Físicas y Matemáticas.

1.1.6.1. Objetivo

Preparar ingenieros para la práctica exitosa de la ingeniería Electrónica en sus diversos campos de aplicación.

1.1.6.2. Ubicación

La Escuela Profesional de Ingeniería Electrónica., está ubicada en Universidad Nacional Pedro Ruiz Gallo, distrito de Lambayeque, Provincia de Chiclayo, Dpto. Lambayeque, Región Lambayeque.

1.1.6.2. Misión

Formar profesionales de alta calidad, caracterizados por tener una sólida formación básica que les permita adaptarse a los rápidos cambios en la tecnología moderna en este nuevo mundo globalizado; con una amplia preparación profesional para resolver los problemas de su especialidad y una formación integral como persona, que les permita convertirse en líderes y conductores de la ciencia y la tecnología, participando en la toma de decisiones que contribuyan al desarrollo del país.

1.1.6.3. Visión

Asumir el liderazgo en la formación de los Ingenieros Electrónicos en el Perú y brindar educación de calidad, comprometida con el desarrollo social-económico sostenido del país, inculcando en los alumnos valores éticos y cultivando su creatividad e innovación.

1.1.6.4. Descripción de sus instalaciones

Las instalaciones de la Escuela Profesional de Ingeniería Electrónica están conformadas por las siguientes principales áreas:

- Laboratorio N° 01:
- Laboratorio N° 02:
- Laboratorio N° 03:
- Laboratorio N° 04:

CAPÍTULO 2

Antecedentes

2.1 ANTECEDENTE 01:

2.1.1. TÍTULO

DISEÑO Y CONSTRUCCIÓN DEL MODELO DE UNA PLANTA DE NIVEL DE LIQUIDOS, SISTEMA VASOS COMUNICANTES.

2.1.2. AUTOR

VICTOR NAPOLEÓN LOPEZ MEDINA

2.1.3. LUGAR

ECUADOR

2.1.4. AÑO

2009

2.1.5. OBJETIVO:

Diseño y construcción del modelo de una planta de nivel de líquido y sistema de vasos comunicantes.

2.1.6. CONCLUSIONES:

- Aprendimos la forma de elaborar un VI en Labview para sintonizar la planta y de esta forma poder obtener las ganancias para el controlador PID, también a identificar la planta y obtener el modelo matemático o función de transferencia de este sistema.

- Cabe recalcar que el sistema fue desarrollado bajo un punto de operación óptimo (25cm) y el control va a estar operando alrededor de ese punto. Cualquier cambio que se realice fuera de este punto de operación el control no va a reaccionar como es debido. La válvula de salida debe estar con una apertura máxima de un 20% para que este disturbio no afecte al control ya que el flujo de salida es menor que el flujo de entrada.

2.1.7. RECOMENDACIÓN:

- Es recomendable para futuro en este proyecto colocar una bomba de mayor potencia, debido a que actualmente se tienen dos bombas con capacidades distintas y de potencia muy baja para poder compensar el flujo de entrada. Haciendo este cambio la respuesta del sistema va a mejorar porque el proceso será más rápido.

2.2. ANTECEDENTE 02:

2.2.1. TÍTULO:

DISEÑO Y CONSTRUCCIÓN DE UNA INTERFAZ DE CONTROL DE NIVEL, TEMPERATURA Y FLUJO DE AGUA EN UN TANQUE PARA USO EN PRÁCTICAS DE LABORATORIO

2.2.2. AUTOR

ESTEBAN RICHMOND SALAZAR

2.2.3. LUGAR

COSTA RICA

2.2.4. AÑO

2009

2.2.5. OBJETIVO

Diseño y construcción de una interfaz de control de nivel, temperatura y flujo de agua en un tanque para uso en prácticas de laboratorio.

2.2.6. CONCLUSIONES

- El transmisor que comunica la instrumentación con el computador personal se ha diseñado basándose en un microcontrolador ATmega16 que limita las señales de entrada de los sensores al ámbito de 0 a 5 VDC, igualmente la salida se limita a una señal de modulación de ancho de pulso de 0 a 5 V; por lo que cualquier instrumentación que se conecte se debe acondicionar a estos valores.

- Se eligen válvulas de solenoide proporcionales, con capacidad nominal de 0 a 4,75×10-5 m3/s, alimentadas por voltaje de corriente directa de 0 a 30 V, por el bajo costo de las mismas; para su operación se ha construido una fuente de energía que suministra como máximo unos 29 VDC, que tras la amplificación de la señal proveniente del transmisor se obtiene un voltaje máximo de 28 VDC.

- Durante la operación de las válvulas PSV se encuentra que presentan un problema de trabamiento para alimentación superior a los 21 V, así como también para voltajes menores a 4 V cuando se introduce un cambio de magnitud relativamente grande. La causa probable de este inconveniente puede ser el daño por desgaste de los empaques o algún tipo de suciedad.

- El flujo máximo obtenido con las válvulas PSV es de 3,33×10-5 m3/s, un 70% del valor nominal, aun cuando el voltaje aplicado es el máximo obtenido por la fuente. Este límite en el flujo real se asocia con el problema de trabado de la válvula, el cual a su vez puede deberse a una obstrucción del obturador causado posiblemente por deterioro de los sellos de las válvulas.

- Los medidores de flujo seleccionados son medidores de micro turbina de rueda que miden en el intervalo de 3,3×10-6 a 8,33×10-5 m3/s de agua, y su respuesta es sumamente lineal frente al flujo, con una ganancia promedio de 1,157×106 %TO/(m3/s) para los tres sensores utilizados.

- Para la medición de nivel se ha seleccionado un sensor de presión manométrica económico debido no solo al costo sino también al ámbito bajo de medición de 0 a 34,5 kPa man, una vez acondicionada la señal se encuentra que la respuesta es altamente lineal en el intervalo de 3 a 45 cm, observándose una no-linealidad a niveles menores a los 3 cm.

2.2.7. RECOMENDACIONES

- Es muy importante sustituir las válvulas PSV o darles mantenimiento a las actuales con el fin de alcanzar el flujo máximo nominal sin inconvenientes por no-linealidades, tales como el trabamiento observado actualmente, puesto que no puede asegurarse que el ámbito de trabajo actual no se vea más limitado con el paso del tiempo.

- Es aconsejable, pero no necesario, instalar un filtro analógico a la señal de salida del sensor de presión, de manera que pueda reducirse el intervalo de control de los lazos de nivel sin inconvenientes por el ruido causado por las vibraciones del soporte. Igualmente sería conveniente utilizar un filtro de paso bajo para la señal que alimenta las válvulas PSV.

- Se sugiere adquirir un adaptador de comunicación USB a serial (RS-232), puesto que la mayoría de los computadores personales más recientes, especialmente los portátiles, no cuentan con el conector DB-9.

- Se recomienda migrar el código fuente del programa a un lenguaje más reciente como el Visual Basic Express 2008 o algún otro lenguaje, a fin de asegurar su compatibilidad con versiones posteriores del MS Windows. Se

sugiere también integrar la ayuda con el código fuente y optimizar la cadena de datos para reducir el tiempo de transmisión.

- Disponer de los acoples necesarios para conectar la entrada de agua y la descarga del tanque principal con el Banco Hidráulico Gunt HM-150, utilizado también con los equipos de prácticas de bombas centrífugas y caída de presión, el cual consta de un tanque de gran capacidad y una bomba centrífuga, o construir un equipo similar. Esto con el propósito de recuperar el agua de salida y mantener el flujo de entrada libre de disturbios por cambios de presión.

- Conectar un segundo sensor de temperatura, basado en transistor, para medir la temperatura de la corriente de entrada, aprovechando el segundo acondicionador de temperatura disponible y uno de los tres canales ADC libres.

2.3. ANTECEDENTE 03:

2.3.1. TÍTULO:

DISEÑO E IMPLEMENTACIÓN DE UNA PLANTA DE NIVEL, CONTROLADA MEDIANTE REDES NEURONALES Y LÓGICA DIFUSA, DESTINADA AL LABORATORIO DE CONTROL INDUSTRIAL DE LA UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE.

2.3.1. AUTOR

JOSÉ CARLOS GARCÉS PICO, JUAN PABLO LEÓN CALDERÓN

2.3.3. LUGAR

ECUADOR

2.3.4. AÑO

2015

2.3.5. OBJETIVO

Realizar el diseño e implementación de una planta de nivel, controlada mediante redes neuronales y lógica difusa, destinada al laboratorio de control industrial de la Universidad de las Fuerzas Armadas ESPE extensión Latacunga.

2.3.6. CONCLUSIONES

- Para el desarrollo exitoso del proyecto, fue primordial recopilar y analizar fundamentos teóricos relacionados al control inteligente, con la finalidad de adquirir una perspectiva clara y objetiva de los temas, dándole un enfoque correcto al diseño y construcción de un controlador inteligente.

- Con la información recopilada de fuentes bibliográficas y digitales se verificó que los controles empleados para este proyecto no requieren de un modelo matemático como su principal característica para la construcción del mismo, por lo que se emplea un lenguaje heurístico, que no es más que la utilización de un lenguaje propio del ser humano, para la toma de decisiones, en el comportamiento y desempeño del control de la planta, es importante notar que este lenguaje debe ser de fácil comprensión y entendimiento para que el programador pueda realizar un buen diseño del controlador.

- Como se ha podido apreciar durante el desarrollo del presente trabajo, se implementaron dos controles inteligentes (fuzzy y neuronal), los mismo que se realizaron en dos etapas, la primera consiste en simular el proceso mediante la plataforma de MATLAB y la otra parte se realizó de manera práctica en LABVIEW, en ambos casos sus respuestas son satisfactorias, ya que mostraron cercanía con las respuestas esperadas tomadas experimentalmente en la planta física.

- Al momento de probar el diseño del controlador difuso en la planta real, ésta se encontró inestable, a tal inconveniente, hubo la necesidad de implementar

un control integrador al diseño, que ayude a eliminar el error entre la variable de proceso y el Set-Point para que el sistema se estabilice.

- En base a la parte experimental, el control Difuso presentó mejor respuesta de la variable de proceso al accionar el actuador, mientras que el control neuronal se mostró lento al arranque de la bomba.

- En la parte real, el control neuronal denotó mayor rapidez que el Difuso, estabilizando su variable de proceso con mayor eficiencia; esto quiere decir, que los puntos designados al entrenamiento de la red fueron los adecuados.

- Si bien el control neuronal presentó mejor tiempo de estabilización que el Difuso, este último brinda menor error entre la variable de proceso y el Set-Point cuando las dos señales se encuentran en estado estable, esto se produce, gracias a la acción integral que tiende a cero a medida que el tiempo transcurre.

- La planta presentó inconvenientes al tomar la señal de voltaje analógico del sensor, a causa de pérdidas de potencial en el conexionado, por lo tanto aquella señal no llegaba a la interface; con aquel problema, se montó un acondicionamiento de señal, donde primero elevó el voltaje original y luego transformó dicha tensión a corriente, evitando pérdidas en el cable.

2.3.7. RECOMENDACIONES

- Las soluciones que se presentó en el control simulado dentro del proceso, son favorables, ya que en la parte práctica a veces se perdía la señal y por ende no existía un control. Por esta razón fue que el control simulado responde de mejor manera a las respuestas esperadas.

- Se recomienda mejorar el diseño de los conjuntos difusos y utilizar un control integrador para eliminar el error en estado estable de la planta y evitar su inestabilidad.

- Se puede mejorar el desempeño de la planta en el arranque del actuador, dotando a la red Neuronal de más puntos de referencia para su entrenamiento y así pueda responder de inmediato cuando la bomba de marcha.

- En base a la experiencia ganada, se podría mejorar el tiempo de estabilización de la planta, al editar el conjunto difuso de salida o la vez su rango de acción, así como también se podría jugar con el integrador y conseguir mejores resultados.

- Se recomienda para la toma de medidas trabajar con un solo tipo de líquido para evitar errores de medida y la constante calibración del sensor ultrasónico.

- Al trabajar con un sensor ultrasónico se dan varios tips tales como: utilizar un acondicionamiento de señal o trabajar con señales digitales incluso señales con ancho de pulsos para evitar pérdidas de potencial en el cable y garantizar una buena señal de control.

2.4. ANTECEDENTE 04:

2.4.1. TÍTULO

CONTROL Y MEDIDA DE NIVEL DE LIQUIDO CON SEÑALES DE ULTRASONIDO

2.4.2. AUTOR

DYLAN ANDRES ALZATE RODRIGUEZ

2.4.3. LUGAR

COLOMBIA

2.4.4. AÑO

2010

2.4.5. OBJETIVO

Controlar el nivel de un sistema hidráulico de almacenamiento de líquido mediante un autómata programable siemens utilizando señales de ultrasonido.

2.4.6. CONCLUSIONES

- Los sistemas de control en lazo cerrado permiten reducir el error que existe entre la señal de referencia y la salida del sistema; de esta manera llevar la salida del sistema a un valor conveniente.

- En la teoría de control, a menudo se usan las funciones de transferencia para caracterizar las relaciones de entrada-salida de componentes o de sistemas que se describen mediante ecuaciones diferenciales lineales e invariantes en el tiempo. A partir del concepto de función de transferencia, es posible representar el sistema mediante ecuaciones algebraicas en función de Laplace.

- Los sensores de ultrasonido son unos de los más económicos, fáciles de manipular y pueden detectar objetos en el orden de los metros sin necesidad de algún filtro o adaptación especial. El SRF05, es un sensor de distancias por ultrasonidos desarrollado por la firma DEVANTECH Ltda. Capaz de detectar objetos y calcular la distancia a la que se encuentra en un rango de 1,7 a 431 cm.

- Una metodología para encontrar los parámetros del controlador, es la respuesta en frecuencia; el cual nos representa la señal de salida en magnitud y fase, facilitando la búsqueda de los controladores, debido a la relación que existe entre el margen de fase y el factor de amortiguamiento relativo.

CAPÍTULO 3

Marco Teórico

3.1 INTRODUCCIÓN

Los procesos industriales no son procesos en estado estable, sino que son dinámicos por naturaleza, los cambios ocurren constantemente y si no se realizan las acciones correctivas apropiadas, las variables importantes del proceso, especialmente aquellas relacionadas con la seguridad, pueden desviarse de los valores de diseño.

El control automático pretende mantener las variables de proceso, temperatura, presión, flujos, composiciones y demás en un valor de operación óptimo (Smith & Corripio, 1997).

El control automático desempeña una función vital en el avance de la ingeniería y la ciencia, y es parte importante e integral de los procesos modernos industriales y de manufactura.

En la actualidad los lazos de control son un elemento esencial para la manufactura económica y próspera de casi cualquier producto, con un enfoque hacia la calidad y constancia en la producción, mejorar el rendimiento y la seguridad, reducción del desperdicio y de energía consumida, además de reducir el trabajo rutinario y aburrido de los operadores (Healey, 1967; Ogata, 1998).

3.2. BREVE HISTORIA DEL CONTROL AUTOMATICO

Uno de los primeros trabajos sobre control de mecanismos, desde la antigüedad, es el control de la razón de flujo para regular un reloj de agua, el cual se reduce al control de nivel del fluido, ya que un pequeño orificio produce un flujo constante cuando la presión es constante.

El mecanismo inventado en la antigüedad para controlar el nivel de un líquido, y todavía existente hoy día, es la válvula flotante semejante a la del depósito de agua de un inodoro convencional. El flotador está hecho de tal manera que, cuando el nivel baja, el caudal hacia el depósito aumenta y cuando el nivel sube, el caudal disminuye y si es necesario se corta (Franklin, Powell, & Emami Naeini, 1991).

Un caso más moderno de retroalimentación es el control de temperatura de un horno para calentar una incubadora, sistema que fue diseñado por Drebbel hacia 1620. En la Figura 3.1 se muestra un croquis de este sistema. El horno consta de una caja que contiene el fuego, con un tubo en la parte superior provisto de un controlador de tiro.

Dentro de la cámara de combustión está la incubadora de paredes dobles y el hueco que queda entre las paredes se llena con agua. El sensor de temperatura es un recipiente de vidrio lleno de alcohol y mercurio colocado en la cámara de agua entre paredes en torno a la incubadora.

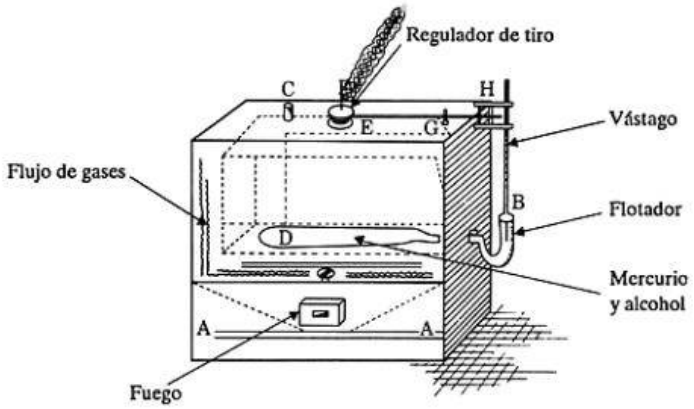


Figura 3.1. Croquis de la Incubadora de Debbel para ampollar huevos de gallina (Franklin et al., 1991, p.5)

A medida que el fuego calienta la caja y el agua, el alcohol se dilata y el vástago con flotador se desplaza hacia arriba, bajando el controlador de tiro sobre la boca del tubo. Si la caja está demasiado fría, el alcohol se contrae, el controlador de tiro se abre y el fuego arde más fuertemente. La temperatura deseada se establece por la longitud del vástago con flotador, que determina la apertura del controlador de tiro para una dilatación específica del alcohol.

En la búsqueda de un medio para controlar la velocidad de rotación de un eje, problema famoso en las crónicas del control automático, como el deseo de controlar automáticamente la velocidad de la piedra de molienda de un molino de viento harinero, de varios métodos intentados, el más prometedor resultó ser el que usaba un péndulo cónico, o controlador de bola flotante. Para medir la velocidad del molino de viento, sus aspas se hacen girar con cuerdas y poleas, para mantener una velocidad fija. Pero fue su adaptación a la máquina de vapor en los laboratorios de James Watt, alrededor de 1788, la que hizo famoso al controlador de bola flotante. La Figura 3.2 muestra un dibujo del diseño de la firma Boulton y Watt de 1788 (Franklin et al., 1991).

Watt determina que una persona controlando la apertura y cierre de las válvulas de vapor no es la mejor manera de mantener la velocidad de su máquina de vapor constante, entonces utiliza el movimiento ascendente de bolas rotatorias como un monitor de velocidad, apagando automáticamente el vapor conforme la velocidad tiende a aumentar y viceversa (Healey, 1967; Ogata, 1998). Watt al ser un hombre práctico no se ocupa de análisis teóricos del controlador; el primer estudio sistemático de la estabilidad del control retroalimentado lo realiza J. C. Maxwell (1868) en su trabajo "On Governors". En este trabajo, Maxwell desarrolla las ecuaciones diferenciales del controlador, linealizándolas en torno al equilibrio, y establece que la estabilidad depende de las raíces de una ecuación característica que tenga partes reales negativas. Maxwell intenta derivar las condiciones de los coeficientes de un polinomio, que se

cumple si todas las raíces tienen una parte real negativa; lo consigue solamente para los casos de segundo y tercer orden.

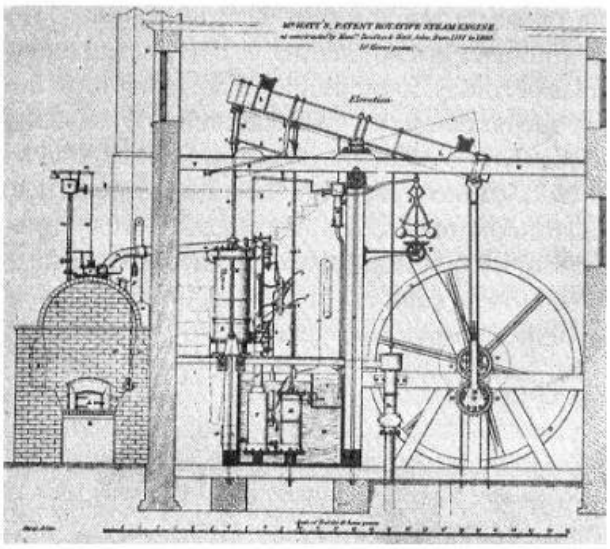


Figura 3.2. Croquis de la máquina de vapor de Watt

Con la introducción de los amplificadores electrónicos, las llamadas a larga distancia llegan a ser posibles en las décadas siguientes a la Primera Guerra Mundial; sin embargo, conforme la distancia aumenta, aumenta también la pérdida de energía eléctrica, requiriéndose más y más amplificadores para reemplazar las pérdidas. Con tantos amplificadores se produce mucha distorsión ya que las pequeñas no linealidades de los tubos al vacío se multiplican una y otra vez; como solución a este problema, en 1927 en los laboratorios de la Bell Telephone H. S. Black propone el amplificador retroalimentado electrónico (Franklin et al., 1991).

Simultáneamente al desarrollo del amplificador retroalimentado, el control retroalimentado de procesos industriales empieza a ser la norma. En este campo, caracterizado por procesos que no solamente son muy complejos sino también no lineales y sujetos a retrasos de tiempo relativamente largos entre actuador y sensor, se desarrolla la práctica del control proporcional, más integral, más diferencial, el controlador PID descrito por Callender, Hartree y Porter (1936). Esta tecnología, basada en un amplio trabajo experimental y aproximaciones linealizadas simples al sistema dinámico, lleva a experimentos estándar apropiados para la aplicación en el campo y finalmente a una satisfactoria "sintonía" de los coeficientes del controlador PID.

Conforme las plantas modernas con muchas entradas y salidas se vuelven más y más complejas, la descripción de un sistema de control moderno requiere de una gran cantidad de ecuaciones. La teoría de control clásica, que trata de los sistemas con una entrada y una salida, pierde su solidez ante sistemas con entradas y salidas múltiples. Desde alrededor de 1960, debido a que la disponibilidad de las computadoras digitales hace posible el análisis en el dominio del tiempo de sistemas complejos, la teoría de control moderna, basada en el análisis en el dominio del tiempo y la síntesis a partir de

variables de estados, se logra desarrollar para enfrentar la creciente complejidad de las plantas modernas y los requerimientos limitativos respecto de la precisión, el peso y el costo en aplicaciones industriales. Ahora que las computadoras digitales son más baratas y más compactas, se usan como parte integral de los sistemas de control. Las aplicaciones recientes de la teoría de control moderna incluyen sistemas ajenos a la ingeniería, como los biológicos, biomédicos, económicos y socioeconómicos. En los años 1980, los descubrimientos en la teoría de control moderna se centran en el control robusto y temas asociados (Ogata, 1998).

La década de los 1990 introduce el concepto de sistemas de control inteligentes, donde una máquina es capaz de lograr un objetivo bajo condiciones de incertidumbre. El control inteligente se basa mucho en el campo de la inteligencia artificial. Se desarrollan las redes neurales artificiales, que se componen de elementos de cómputo simples que operan en paralelo como un intento de emular sus contrapartes biológicas. Así mismo, los controladores de lógica difusa ofrecen control robusto sin la necesidad de un modelo dela dinámica del sistema (Burns, 2001).

3.3. SISTEMA DE CONTROL

Un sistema o proceso está formado por un conjunto de elementos relacionados entre sí, que producen señales de salida en función de señales de entrada. Las variables que afectan un proceso se clasifican en entradas, que denota el efecto de los alrededores sobre el proceso, y salidas, que denota el efecto del proceso sobre los alrededores.

3.3.1. Características

Las principales características que se deben buscar en un sistema de control serán:

- Mantener el sistema estable, independiente de perturbaciones y desajustes.
- Conseguir las condiciones de operación objetivo de forma rápida y continua.
- Trabajar correctamente bajo un amplio abanico de condiciones operativas.
- Manejar las restricciones de equipo y proceso de forma precisa.

3.3.2 Importancia

La implantación de un adecuado sistema de control de proceso, que se adapte a las necesidades de determinado sistema, significará una sensible mejora de la operación. Principalmente los beneficios obtenidos serán:

- Incremento de la productividad
- Mejora de los rendimientos
- Mejora de la calidad
- Ahorro energético

- Control medioambiental
- Seguridad operativa
- Optimización de la operación del proceso/ utilización del equipo
- Fácil acceso a los datos del proceso

El bucle de control típico estará formado por los siguientes elementos, a los que habrá que añadir el propio proceso.

3.3.3. Estructura de un sistema de control

Sistema: es la combinación de componentes que actúan conjuntamente y cumplen un determinado objetivo.

Variable de entrada: es una variable del sistema tal que una modificación de su magnitud o condición puede alterar el estado del sistema.

Variable de salida: es una variable del sistema cuya magnitud o condición se mide.

Perturbación: es una señal que tiende a afectar el valor de la salida de un sistema. Si la perturbación se genera dentro del sistema se la denomina interna, mientras que una perturbación externa se genera fuera del sistema y constituye una entrada.

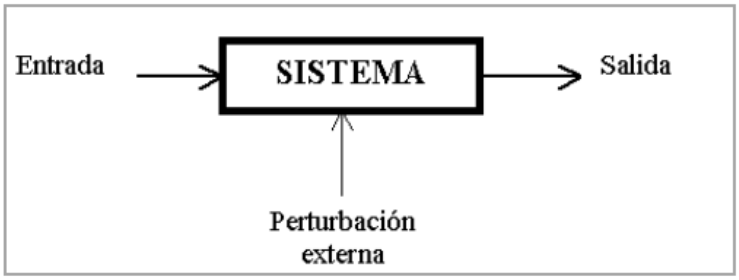


Figura 3.3. Estructura de un Sistema de Control

3.3.4. Características de un sistema de control

Las principales características que se deben buscar en un sistema de control serán:

1. Mantener el sistema estable, independiente de perturbaciones y desajustes.
2. Conseguir las condiciones de operación objetiva de forma rápida y continua.
3. Trabajar correctamente bajo un amplio abanico de condiciones operativas.
4. Manejar las restricciones de equipo y proceso de forma precisa.

3.3.5 Lazos de Control

3.3.5.1 Generalidades

El bucle de control típico estará formado por los siguientes elementos, a los que habrá que añadir el propio proceso.

- **Elementos de medida (Sensores):** Generan una señal indicativa de las condiciones de proceso.

- **Elementos de comparación (Controladores):** Leen la señal de medida, comparan la variable medida con la deseada (punto de consigna) para determinar el error, y estabilizan el sistema realizando el ajuste necesario para reducir o eliminar el error.

- **Elementos de ajuste (Válvulas y otros elementos finales de control):** Reciben la señal del controlador y actúan sobre el elemento final de control, de acuerdo a la señal recibida.

La serie de operaciones de medida, comparación, cálculo y corrección, constituyen un ciclo cerrado. El conjunto de elementos que hacen posible este control reciben el nombre de lazo de control (control loop).

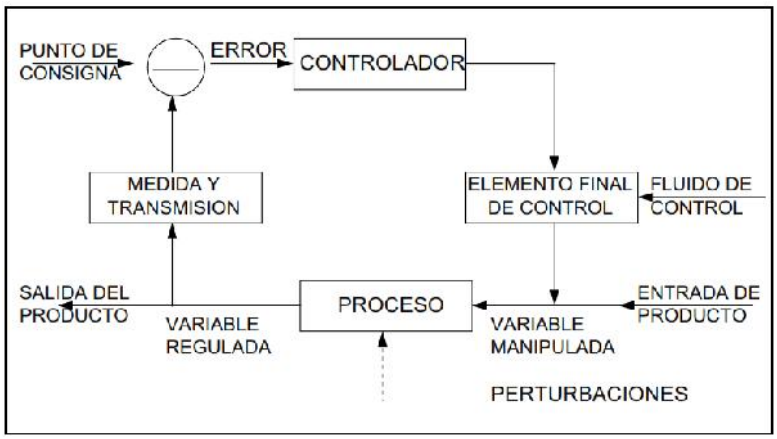


Figura 3.4. Lazos de Control

3.3.5.2. Lazo abierto

A lazo abierto, la variable de proceso no es comparada y se genera una acción independientemente de las condiciones de la misma.

Son sistemas de control en los que la salida o resultado del proceso no tiene ningún efecto sobre la acción de control, es decir, en un sistema de control de lazo abierto la salida no se mide (no se retroalimenta) para comparar con lo que deseamos obtener y así verificar cual es el error.

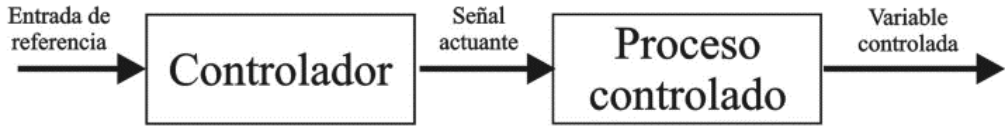


Figura 3.5. Diagrama a bloques de un Sistema a Lazo Abierto

Las ventajas de los sistemas de control de lazo abierto son:

- Montaje simple y facilidad de mantenimiento.
- Más económico que un sistema de lazo cerrado equivalente.
- No hay problemas de estabilidad.
- Es conveniente cuando es difícil económicamente medir la salida.

Las desventajas que tienen dichos sistemas son:

- Las perturbaciones y las modificaciones en la calibración introducen errores, y la salida puede diferir de la deseada.
- Para mantener la calidad necesaria a la salida, puede ser necesario efectuar periódicamente una re calibración.

3.3.5.3. Lazo cerrado

Son aquellos en los que la señal de salida tiene efecto directo sobre la acción de control, esto es, los sistemas de control de lazo cerrado son sistemas de control retroalimentados.

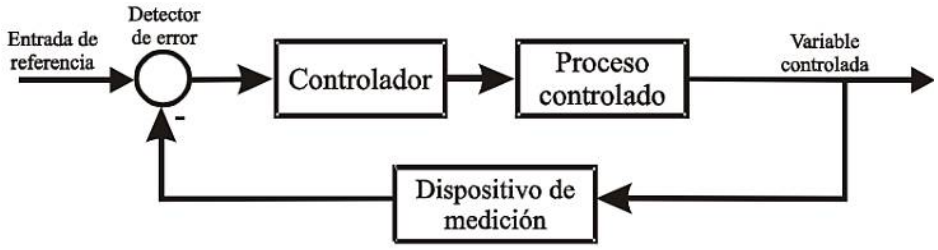


Figura 3.6. Diagrama a bloques de un Sistema a Lazo Cerrado

A diferencia del control de lazo abierto, en el de lazo cerrado sí se mide la salida del proceso para verificar si está dentro del valor deseado al compararlo con éste.

3.3.5.4. Componentes de lazo de control

Los elementos principales de un lazo de control pueden clasificarse de la forma siguiente:

- **Sistema de medición:** son los elementos que se utilizan para determinar y comunicar al sistema de control el valor de la variable controlada, o variable de proceso.
- **Elemento de control:** es la respuesta del sistema de medición que puede ser una señal eléctrica, neumática o visual.
- **Sistema de control:** son los elementos del controlador relacionados con la generación de la acción correctiva.
- **Unidad de potencia:** es la parte del sistema de control que aplica energía para accionar el elemento final de control.
- **Elemento final de control:** es la parte del sistema de control que modifica directamente el valor de la variable manipulada.

3.3.6 Formas de control

Aunque existen variantes sobre alguno de los métodos de control que se describen, los más utilizados son los siguientes:

3.3.6.1. Control de dos posiciones

Comúnmente llamado control Todo-Nada, On-Off.

Como su nombre indica, en el control de dos posiciones el elemento final de control solo ocupa una de las dos posiciones posibles.

Es usado a menudo, cuando existe grandes capacidades y la energía entrante y saliente es pequeña comparada con la capacitancia del sistema.

Las válvulas solo podrán asumir dos estados:

- ✓ Totalmente abiertas (ON)
- ✓ Totalmente cerradas (OFF)

Es la forma de control más simple y más económico.

Un ejemplo simple es donde se quiere controlar la temperatura y el nivel del tanque que se muestra en la siguiente figura:

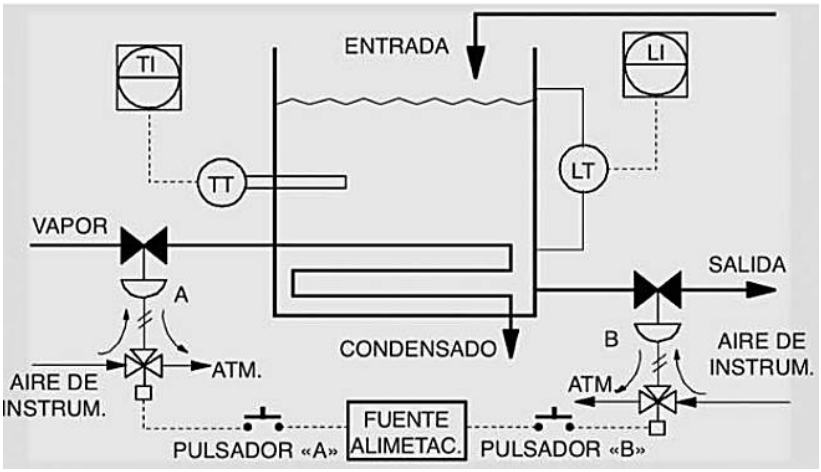


Figura 3.7. Control de dos posiciones

Para ello se realiza el siguiente algoritmo de control:

E= SP – PV

Si E>0 la salida será igual a 100%
Si E<0 la salida será igual a 0%

Siendo: **E=** ERROR
SP= SET POINT
PV=VARIABLE DE PROCESO

La posición de la válvula (V) será, por tanto: ***V=f (signo E)***

CERRADA, cuando **E** es negativo.

ABIERTA, cuando **E** es positivo.

La salida de control entonces es:

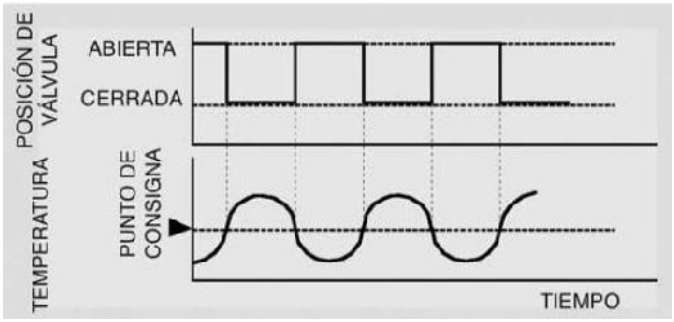


Figura 3.8. Salida de Control

Banda muerta

El intercambio de calor no se realiza instantáneamente, sino que transcurre un tiempo para que la energía se transfiera al líquido en el tanque o deje de transferirse.

La temperatura oscila entre un rango de valor por encima o por debajo del punto de referencia o set point.

La amplitud de la oscilación dependerá del retardo del proceso y la atención que el operador preste para determinar el momento en que realiza el cambio de posición de la válvula.

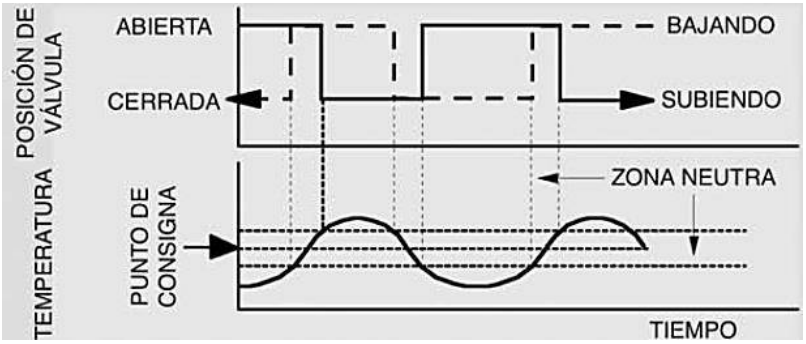


Figura 3.9. Diseño de la Banda Muerta

El diseño de la banda muerta, también llamada histéresis, previene que la salida no conmute rápidamente de “OFF” a “ON”.

Si la histéresis esta seteada en un rango muy estrecho la salida comenzaría a cambiar de estado tan rápido que producirá una disminución del tiempo de vida útil de algún rele o contacto y, además, la elevación de temperatura en los componentes; por lo tanto esta histéresis debería estar seteada con un suficiente tiempo de retardo para evitar esta condición.

3.3.6.2. Control proporcional

El control de dos posiciones produce variaciones y un ciclo continuo que muchos procesos no pueden tolerar. El control proporcional modula al elemento final de control en forma continua entre los límites máximos y mínimos y es la base de trabajo de la mayoría de los reguladores actuales.

Con el control proporcional la salida del controlador es directamente proporcional a su entrada; la entrada es la señal de error, e, la cual es una función del tiempo. De esta manera:

$$m = Kp.e + m_o$$

Este modo de control existe un error inherente, para reducir o eliminar este error, se necesita el parámetro denominado reset manual (m_o), también se le conoce como **bi♦as**.

La salida del controlador depende sólo de la magnitud del error en el instante en el que se considera. La función de transferencia, $G_c(s)$ para el controlador es, por lo tanto:

$$G_c(s) = K_p$$

El controlador es, en efecto, sólo un amplificador con una ganancia constante. En cierto tiempo, un error grande produce una salida grande del controlador. La ganancia constante, sin embargo, tiende a existir sólo sobre cierto rango de errores que se conoce como banda proporcional. Una gráfica de la salida contra el error sería una línea recta con una pendiente de K p en la banda proporcional.

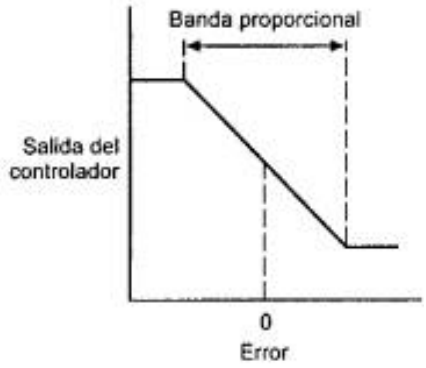


Figura 3.10. Salida del Controlador

Es común expresar la salida del controlador como un porcentaje de la posible salida total de éste. De este modo, un 100% de cambio en la salida del controlador corresponde a un cambio en el error desde un extremo a otro de la banda proporcional. Así

$$BP = \frac{100}{Kp}$$

Debido a que la salida es proporcional a la entrada, si la entrada al controlador es un error en la forma de un escalón, entonces la salida es también un escalón, y es exactamente una versión a escala de la entrada dentro de su banda proporcional.

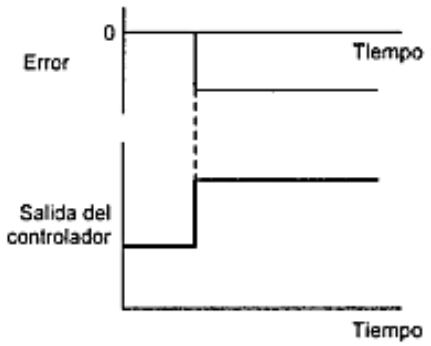


Figura 3.11. Escala de la entrada dentro de su Banda Proporcional

El control proporcional es sencillo de aplicar, en esencia sólo se requiere alguna forma de amplificador. Éste podría ser un amplificador electrónico o un amplificador mecánico.

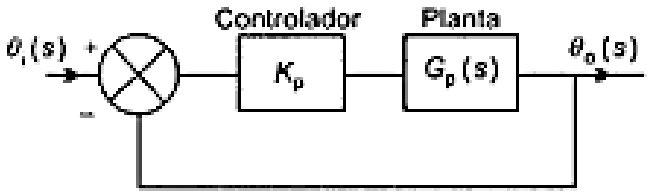


Figura 3.12. Control Proporcional

El resultado es una función de transferencia en lazo abierto:

$$G_0(s) = K_p G_p(s)$$

El resultado es una función de transferencia en lazo cerrado:

$$G_0(s) = \frac{K_p G_p(s)}{1 + K_p G_p(s)}$$

3.3.6.3 Control integrativo

Con el control integral la salida del controlador es proporcional a la integral de la señal de error e con el tiempo, es decir:

$$salida(\mathrm{s}) = K_i \int_0^t e dt$$

Donde K_i es la constante denominada ganancia integral. La integral entre 0 y t es, de hecho, el área bajo la gráfica del error entre 0 y t. Así, debido a que después de que el error comienza, el área se incrementa en una razón regular, la salida del controlador se debe incrementar en una razón regular. La salida en cualquier tiempo es, entonces, proporcional a la acumulación de los efectos de los errores pasados.

Al tomar la transformada de Laplace de $salida\,(s)/\,e(s)$ da por resultado la función de transferencia, para el controlador integral, de:

$$\mathrm{G}_c(\mathrm{s}) = \frac{K_i}{s}$$

Así, para el sistema de la forma:

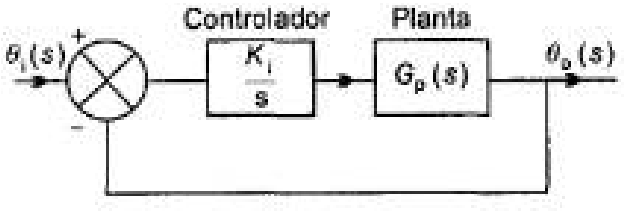


Figura 3.13. Control Integrativo

Se tiene una función de transferencia en lazo abierto de:

$$\mathrm{G}_0(\mathrm{s}) = \left(\frac{K_i}{s}\right)\mathrm{G}_p(\mathrm{s})$$

3.3.6.4. Control derivativo

Con la forma derivativa del controlador, la salida del controlador es proporcional a la razón de cambio con el tiempo del error e, es decir:

$$salida(\mathrm{s}) = K_d\,\frac{de}{dt}$$

Donde K_d es la ganancia derivativa y tiene unidades de s. Con el control derivativo, tan pronto como la señal de error inicia puede haber una salida del controlador muy grande, puesto que ésta es proporcional a la razón de cambio de la señal de error y no a su valor. De este modo puede proporcionar una acción correctiva grande antes de que se presente un error grande en realidad. Sin embargo, si el error es constante, entonces no hay acción correctiva, aun si el error es grande. Así, el control derivativo es insensible a señales de error constantes o que varían con lentitud y, en consecuencia, no se usa solo, sino combinado con otras formas de controlador.

Al tomar la transformada de Laplace de $salida\ (s)/\ e(s)$ resulta, para el control derivativo, una función de transferencia:

$$\mathrm{G}_c(\mathrm{s}) = K_d s$$

Por lo tanto, para el sistema en lazo cerrado

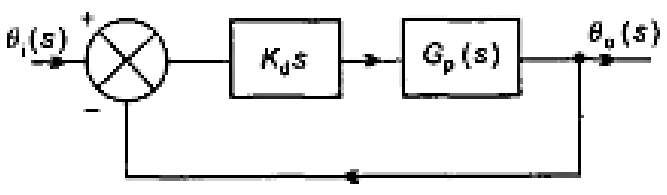


Figura 3.14. Control Derivativo

La presencia del control derivativo produce una función de transferencia en lazo abierto de

$$\mathrm{G}_0(\mathrm{s}) = \frac{K_d s \mathrm{G}_p(\mathrm{s})}{1 + K_d s \mathrm{G}_p(\mathrm{s})}$$

Como se mencionó, la acción derivativa no se usa sola sino sólo en conjunto con otra forma de controlador. Cuando se usa esta acción de control se logra que la respuesta sea más rápida.

El control solo proporcional es adecuado en donde los cambios de carga son relativamente pequeños y en donde el control preciso no es crítico. Si se necesita un control preciso, la desviación u offset inherente al control proporcional, determina que se requieran funciones adicionales para lograr un mejor control.

3.3.6.5. Control proporcional integrativo (PI)

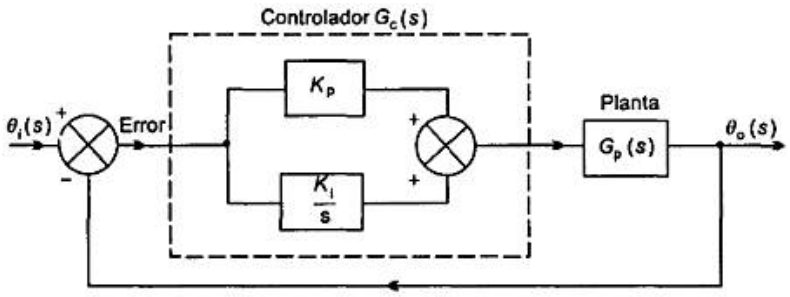


Figura 3.15. Control Proporcional Integrativo (PI)

La reducción en la estabilidad relativa como resultado de usar el control integral se puede resolver, como una extensión, mediante el control proporcional integral (PI). Para tal combinación la salida del controlador es:

$$salida(\mathrm{s}) = K_p e + K_i \int_o^t e dt$$

El tipo de salida del controlador que se presenta con dicho sistema cuando existe una entrada de error tipo escalón es representado de la siguiente forma:

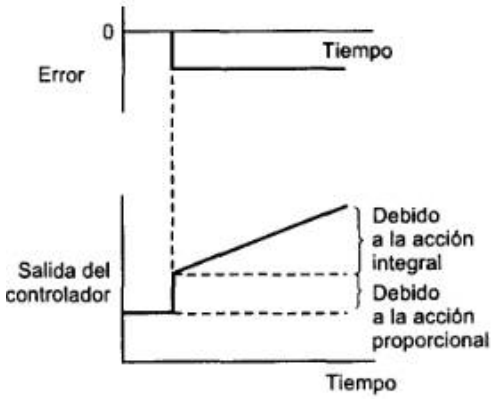


Figura 3.16. Salida del Control PI

Al tomar la transformada de Laplace de la ecuación $salida\ (s)/\ e(s)$ se obtiene una función de transferencia, para el controlador PI de:

$$G_c(s) = \frac{K_p\left[s+\frac{K_i}{K_p}\right]}{s} \qquad G_c(s) = \frac{K_p\left[s+\frac{1}{\tau_i}\right]}{s}$$

En consecuencia, la función de transferencia de la trayectoria directa para el sistema es:

$$G_0(s) = \frac{K_p\left[s+\frac{1}{\tau_i}\right]G_p(s)}{s}$$

3.3.6.6. Control proporcional derivativo (PD)

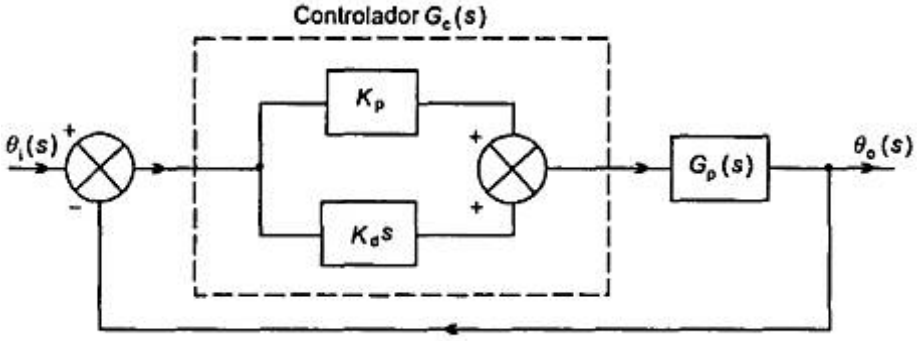


Figura 3.17. Control Proporcional Derivativo (PD)

Si el control derivativo se usa con el control proporcional, entonces la función de transferencia en lazo abierto se convierte en

$$G_0(s) = K_p[(1/\tau_d) + s])G_p(s)$$

Donde $\tau_d = K_P/K_d$ y se denomina constante de tiempo derivativa.

3.3.6.7. Control proporcional integral derivativo (PID)

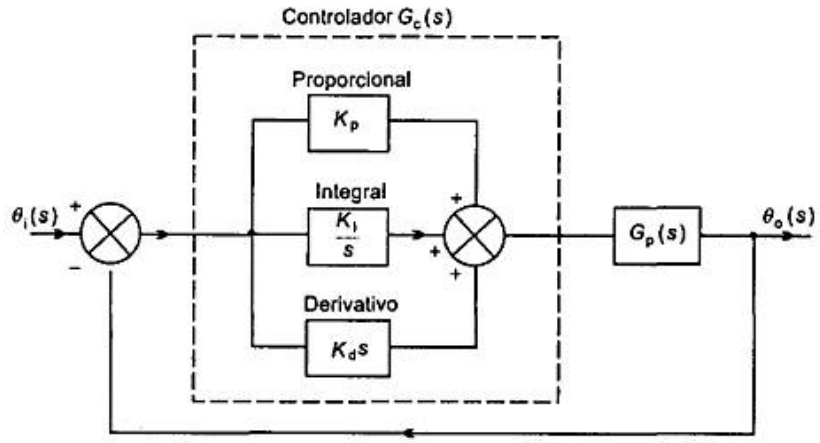


Figura 3.18. Control Proporcional Integral Derivativo (PID)

El controlador proporcional integral derivativo (PID), mejor conocido como controlador de tres términos, dará una salida, para una entrada de error e, de:

$$salida\ (s) = K_P e + K_i \int_0^t edt + K_d \frac{de}{dt}$$

La función de transferencia, $salida\ (s)/\ e(s)$, del controlador es, de esta manera:

$$G_c(s) = K_P + \frac{K_i}{s} + K_d s$$

Debido a que la constante de tiempo integral, τ_i, es $K_p/\ K_i$ y la constante de tiempo derivativa, τ_d, es $K_d/\ K_p$ la ecuación $G_c(s)$ se puede escribir como:

$$G_c(s) = K_P \left(1 + \frac{1}{\tau_i s} + \tau_d s \right)$$

La función de transferencia en lazo abierto para este sistema es:

$$G_0(s) = \frac{K_P(\tau_i s + 1 + \tau_i \tau_d s^2) G_p(s)}{\tau_i s}$$

3.3.6.8 Control en cascada

En la práctica existe gran cantidad de procesos con lazos simples de control feedback entre los que se produce interacción. Para eliminarla es necesario medirla y controlarla por medio de un sistema de control denominado cascada.

Se muestra un ejemplo típico a partir del cual se va a contemplar el funcionamiento del sistema:

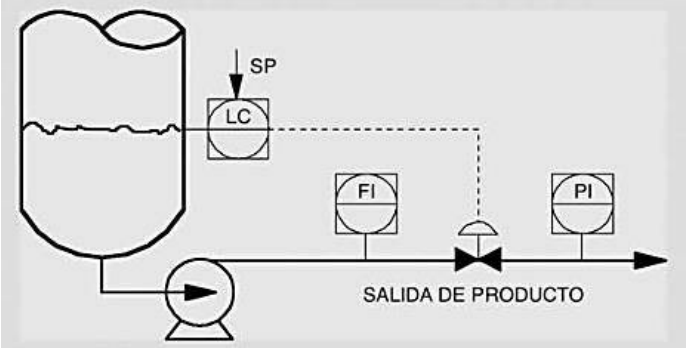


Figura 3.19. Funcionamiento del Sistema

Para controlar el nivel en el fondo de un recipiente se utiliza, en principio, un controlador que actúa directamente sobre la válvula. Mientras aguas debajo de la válvula automática no se produzca ninguna perturbación se efectuara un control de nivel aceptable. Si por alguna causa disminuye la presión aguas abajo, la válvula dejara pasar más caudal para la misma apertura, al haber aumentado la diferencia de presión a través de ella. La posición de válvula se corregirá cuando disminuya el nivel y su controlador disminuya la salida en cantidad suficiente para buscar un nuevo equilibrio. En otras palabras, existe interacción entre caudal y nivel, la cual se puede amortiguar efectuando control en cascada.

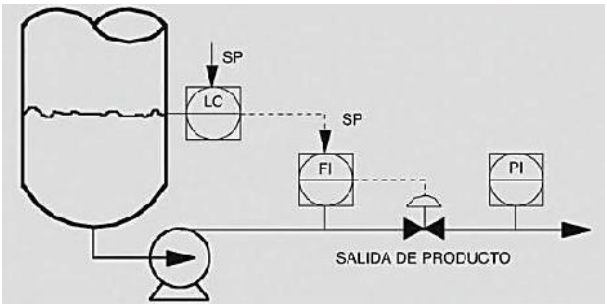


Figura 3.20. Control en Cascada

Con esta técnica, tan pronto se modifique el caudal se llevara a cabo la corrección necesaria sobre la válvula antes que se vea afectado el nivel del recipiente.

En este punto se puede ver que un sistema de control en cascada utiliza dos controladores feedback, uno llamado primario y el otro secundario. El

controlador de nivel (primario), se utiliza para fijar el set point del secundario. La variable a controlar es la medida del controlador primario, mientras que la medida de caudal del secundario es una variable intermedia.

En un sistema de control en cascada, la dinámica del lazo secundario debe ser siempre más rápida que la del primario.

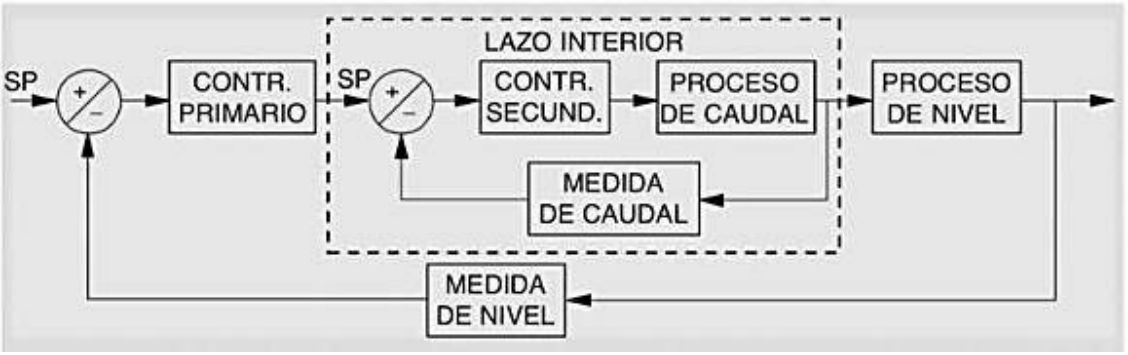


Figura 3.21. Dinámica del Lazo secundario frente al Lazo Primario

A partir de este diagrama de bloques del ejemplo mostrado, se puede deducir dos comportamientos:

- Si la perturbación ocurre en el lazo de control interior (caudal), el controlador secundario inicia la acción correctiva antes que se traslade al lazo de control exterior (nivel).
- Si la perturbación ocurre en el lazo exterior (nivel), el comportamiento de la cascada hace que se modifique el set point del lazo interior. En este caso el conjunto se comporta prácticamente como si fuera un solo control feedback.

3.4. ELEMENTOS DE MEDICIÓN Y TRANSMISIÓN

3.4.1 Variables de Proceso

Las variables tradicionales que se miden y controlan en los procesos son cuatro: Presión, Temperatura, Nivel de interface y Caudal. Estas variables están vinculadas a las condiciones operativas de los procesos. También interesa en la industria de procesos ciertas características físicas (densidad, viscosidad, etc.) y químicas (composición, conductividad, pH, etc.) que también se miden y controlan, pero en mucha menor escala.

3.4.2. Elementos de medición y transmisión

Son los dispositivos que se encargan de transformar la variable de ingeniería (temperatura, por ejemplo) en una señal mecánica, eléctrica, etc. que puede ser usada por otros instrumentos (indicadores, controladores, registradores, etc.). Estos dispositivos tienen dos partes:

- Elemento primario: es el que capta la variable a medir y produce cambios en propiedades físicas que luego puede transformarse en una señal.

- Elemento secundario: capta la señal elaborada por el elemento primario y la transforma en una salida (indicación por ejemplo) o genera una señal estandarizada que puede ser captada por otro instrumento en forma local o remota.

Estas dos partes pueden estar claramente separadas como en el caso de un tubo Venturi (elemento primario) con transmisor de presión diferencial (elemento secundario) o bien ambos elementos están confundidos en un mismo dispositivo (medidor de presión tipo Bourdon con indicación de aguja).

Analizando las relaciones causa efecto, se puede representar a un medidor-transmisor como dos sistemas en serie:

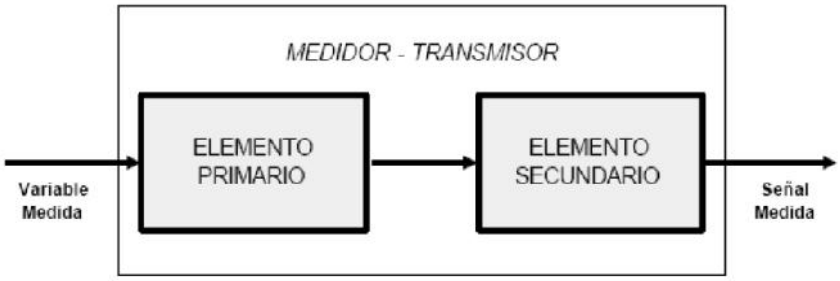


Figura 3.22. Representación de un Medidor – Transmisor

3.5. MEDICIÓN DE NIVEL

El nivel es una variable muy importante en los procesos ya que está vinculada a la operación del equipo, al inventario, la logística y los costos. Lo más común es designar con nivel a la posición de la interface líquido-gas o sólido-gas.

Pero también se suele medir y controlar la interface líquido-líquido y líquido-sólido.

No existe algo así como "un medidor universal" que sea aplicable a todos (o la mayoría) de los casos. Cada situación debe ser cuidadosamente analizada, ya que existe un sinnúmero de condiciones a tener en cuenta como tipo de sólidos o fluido, agresividad física o química, existencia de espuma, ángulos de talud en sólidos, etc.

En la actualidad existe una gama enorme de tecnologías de medición de nivel. Se comentan algunas de las tecnologías más comunes para la medición de nivel.

3.5.1. Tipos de medición de nivel

La medición de nivel podemos clasificarla de dos tipos continua y límites de nivel.

3.5.1.1 Medidores Continuos

Consiste en monitorear en tiempo real el nivel del tanque de almacenamiento.

- **RADIOMÉTRICO:**

  Se utiliza tanto para medición continua o discreta. El transmisor no entra en contacto con el material ni con el recipiente, tanto fuera o dentro de él. Se utiliza en contenedores de materiales o líquidos inflamables, venenosos o agresivos. Este dispositivo puede ser encontrado en tanques de ácido, hervidores, silos de cemento, ciclones, hornos rotativos, agitadores y mezcladores. Debido a que la fuente sola emite rayos gamma, el material y el tanque pueden ser contaminados radiactivamente.

  Principio de funcionamiento:

  La fuente de rayos gamma, tanto de componentes de cesio o cobalto, emite radiación que es atenuada a medida que pasa a través de los materiales. Un detector, montado en el lado opuesto del recipiente, convierte esta radiación en una señal eléctrica. La amplitud de la señal es determinada por la distancia entre la fuente gamma y el detector y también por el ancho y la densidad del material. El ancho y las paredes del recipiente, cuya atenuación de la radiación es constante, se tienen en cuenta en el cálculo de la señal. La determinación del nivel, se basa en la absorción de radiación por el producto que contiene el tanque.

Figura 3.23. Transmisor de Nivel: Radiométrico

- **ULTRASONICO**

  Se usan para la medición continua de nivel, suelen montársela través de la parte superior del recipiente o tanque.

  Principio de funcionamiento:

  Consiste en emitir un pulso de energía que viaja a la velocidad del sonido en el espacio de vapor que se encuentra por encima del líquido o polvo. La señal es reflejada por la superficie del líquido o polvo y va de vuelta al receptor. Se mide el

tiempo entre la señal emitida y la señal recibida. A partir de esa medición de tiempo y con la velocidad del sonido en el vapor se calcula la distancia desde el receptor a la superficie del líquido o polvo. En los últimos años, los medidores sónicos de nivel han mejorado en exactitud cuándo se los aplica en forma adecuada.

Figura 3.24. Transmisor de Nivel: Ultrasónico

- **HIDROSTÁTICO:**

  Es utilizado para medición continua de nivel en tanques que contengan líquidos o barros, en la industria química, farmacéutica y alimenticia, como también en tratamiento de agua y aguas residuales. La sonda, formando un sensor de presión se encuentra de distintos diseños de construcción para diversas aplicaciones, por ejemplo: para ser montadas a un costado del tanque, o arriba, para materiales corrosivos, etc.

  Principio de funcionamiento:

  El peso de una columna de líquido genera una presión hidrostática. A densidad constante, la presión hidrostática es solamente función de la altura de la columna de líquido:

$$\rho_{hidrostática} = \rho \,.\, g \,.\, h$$

  Se usan para la medición continua de nivel, suelen montársela través de la parte superior del recipiente o tanque.

Figura 3.25. Transmisor de Nivel: Hidrostático

- **MAGNETOSTRICTIVOS:**

  El medidor de nivel NMT de Kobold es un sensor contador controlado por flotador muy exacto para niveles de sensado continuo. El instrumento comprende dos partes:

  - ✓ Sensor magnetostrictivo en el tubo de medición.
  - ✓ Transmisor de cuatro hilos en la caja de conexión.

  Principio de Funcionamiento

  El principio de medición se basa en la medida del tiempo de eco. Un alambre magnetostrictivo está tensado en el tubo de medición. Los pulsos de corriente se transmiten a través del alambre, generando así un campo magnético anular alrededor del alambre. El alambre también es magnetizado axialmente por imanes acondicionados en el flotador. Debido a la superposición de ambos campos magnéticos, un impulso torsional se genera en la vecindad del imán del flotador, que se propaga con velocidad ultrasónica en ambas direcciones. La distancia del imán de flotador a un punto cero definido se mide con una medida del tiempo de eco. Los sistemas electrónicos integrados transforman la señal a una señal analógica estandarizada.

Figura 3.26. Transmisor de Nivel: Magnetostrictivos

- **RADAR:**

  Los medidores radar a dos hilos ofrecen las ventajas de no tener partes móviles y no haber contacto con el líquido que se mide. La medición no se ve afectada por la temperatura, presión y densidad del producto. Para una planta de proceso, las cualidades adicionales de Radar disponibles en el transmisor permiten conseguir una mayor productividad, reducir el coste de la propiedad y aumentar la disponibilidad del proceso. Se reducen los gastos de ingeniería e instalación así como los gastos de explotación y mantenimiento.

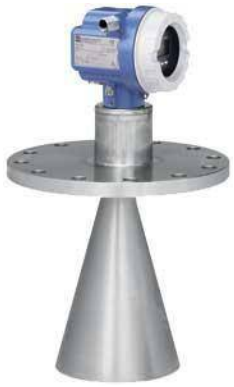


Figura 3.27. Transmisor de Nivel: Radar

3.5.1.2. Interruptores de Nivel

Nos permiten saber el estado del nivel como mínimo y máximo.

- **PALETA ROTATIVA:**

  Los monitores de nivel de paleta rotativa sirven como interruptores límite para los materiales macizos, polvorientos, y granulados. Son convenientes para uso con pesos de materiales macizos desde 0,3 a 2,5 t/m y tamaños de partícula hasta 50 milímetros. Diversas posiciones de instalación (horizontal, vertical, inclinada) así como un amplio rango de modelos permiten el uso de los monitores de nivel de paleta de KOBOLD para casi todas las aplicaciones.

  Método de Operación

  Un motor síncrono que gira alrededor de cierto ángulo en una extensión de eje está sujetado a un tope extremo mediante un resorte. El motor maneja una paleta rotativa que sale a un recipiente por medio de un eje. Tan pronto el llenado alcanza las paletas, su rotación se obstaculiza y se detiene.

  El torque de reacción gira el motor y hace operar un microcontacto (contacto N/A). El motor se apaga con un segundo interruptor. Si el nivel baja, la paleta rotativa se libera y

el motor retorna a su posición original mediante el resorte. Esto enciende el motor de nuevo y el contacto se conmuta a su posición anterior.

Figura 3.28. Interruptor de Nivel: Paleta Rotativa

- **MEMBRANA:**

  Los monitores de nivel de la membrana permiten control económico del nivel de productos macizos en recipientes de almacenaje. Pueden ser utilizados para indicar estados llenos y vacíos y demanda de la carga para productos macizos, polvoriento y granulado. Son convenientes para el uso con materiales macizos (0,3 a 2,5 t/m) y tamaños de partícula de hasta 30 mm. Los dispositivos funcionarán sin fallas con la condición de que los materiales macizos fluyan fácilmente a un ángulo no demasiado pequeño. Solamente tales materiales ejercen la presión suficiente de operación en el detector acondicionado en la pared del silo.

  Método de Operación

  La cubierta hecha de fundición de aluminio o de plástico reforzado con fibra de vidrio lleva la membrana retendida por un anillo de atornillamiento. Con su propio peso, el material macizo hace presión contra la membrana, la que es pre-tensionada con un resorte a través del soporte.

  Un émbolo fijado a la membrana transfiere la presión directamente a un microcontacto con contacto changeover. Si se desploma el material macizo, se releva la membrana y el contacto se conmuta a donde estaba.

  La sensibilidad se puede ajustar con un resorte. El monitor se puede optimizar así para el tipo de relleno y las condiciones de instalación.

Figura 3.29. Interruptor de Nivel: Membrana

- **MAGNÉTICO:**

  Los interruptores de nivel magnéticos se utilizan para monitorear y controlar niveles de líquidos en recipientes. Los interruptores de nivel magnéticos se fabrican a la especificación del cliente. Una descripción de los tipos disponibles con longitudes mínimas de tubos de medición se precisa en las páginas siguientes. Refiérase por favor a esta descripción al poner su orden. Además cualquier límite se puede especificar dentro de los límites especificados en el folleto.

  Método de Operación

  Los interruptores de flotador magnéticos están equipados con un contacto herméticamente sellado que se sitúa en el tubo. El flotador que se desliza, en el tubo contiene un imán de anillo cuyo campo magnético conmuta el contacto sellado de una manera no contactante. Los contactos sellados están disponibles como N/A, N/C o contacto changeover. El flotador que se desliza hacia arriba y hacia abajo en el líquido es la única pieza móvil en los interruptores de flotador magnéticos.

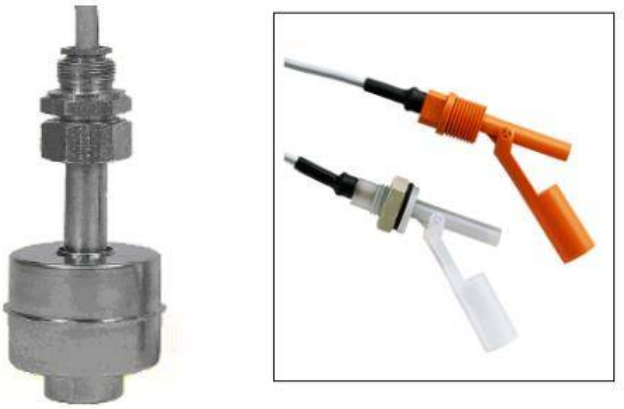


Figura 3.30. Interruptor de Nivel: Magnético

- **FLOTADORES:**

  Los niveles de líquidos se pueden monitorear fácilmente con los siguientes tipos de interruptores de flotador. Los esquemas de control de nivel se pueden implementar con por lo menos dos flotadores, donde uno funciona como contactor mínimo, y el otro como contactor máximo. Los interruptores se adecuan para aplicaciones donde los interruptores de nivel magnético son inadecuados debido al peligro de atoramiento de flotador con partículas o depósitos de suciedad. Dependiendo de la forma y el material usado, También se pueden monitorear con interruptores de flotador los medios extremadamente agresivos, calientes, sucios, pastosos.

  Descripción

  El flotador contiene un cilindro hueco o una bola con un interruptor de mercurio integrado o microcontacto. El contacto se provee como contacto changeover; puede ser conectado como un contacto N/A o contacto N/C como opción. El contacto cambia cuando el líquido pasa sobre o bajo la posición de flotador horizontal. El punto de interrupción es fijado por la instalación lateral del interruptor en la posición deseada o sujetando el cable con abrazaderas. El punto de interrupción se fija usando pesas cuando está instalado en la parte superior.

Figura 3.31. Interruptor de Nivel: Flotador

3.6. ACTUADORES

Un actuador es un dispositivo capaz de transformar energía hidráulica, neumática o eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado. Este recibe la orden de un regulador o controlador y en función a ella genera la orden para activar un elemento final de control como, por ejemplo, una válvula.

Cada uno de los sistemas presentan características diferentes, entre las mas importantes a considerar son:

- Potencia
- Controlabilidad
- Peso y Volumen
- Precisión
- Velocidad
- Mantenimiento
- Costo

3.6.1. Tipos de Actuadores

3.6.1.1. Actuadores Neumáticos

A los mecanismos que convierten la energía del aire comprimido en trabajo mecánico se les denomina actuadores neumáticos. Aunque en esencia son idénticos a los actuadores hidráulicos, el rango de compresión es mayor en este caso, además de que hay una pequeña diferencia en cuanto al uso y en lo que se refiere a la estructura, debido a que estos tienen poca viscosidad.

Figura 3.32. Actuador Neumático

Existen dos tipos:

a) Cilíndricos Neumáticos

En este tipo de actuador el desplazamiento se consigue como la consecuencia de la diferencia de presión en ambos lados del cilindro sobre un embolo.

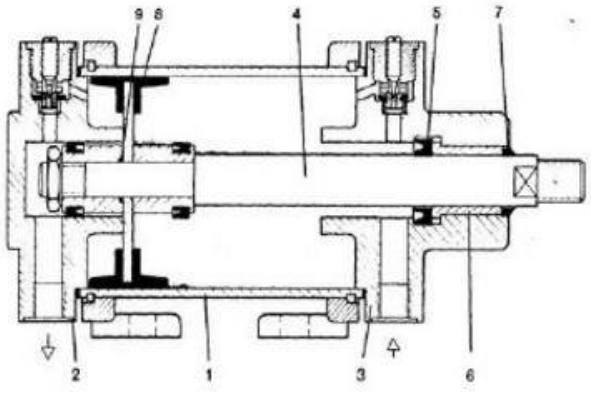


Figura 3.33. Actuador Cilíndrico Neumático

Existen dos clases:

- **Efecto Simple**

  En este tipo el embolo se desplaza en un sentido como resultado del empuje ejercido por el aire a presión, mientras que en el otro sentido desplaza como consecuencia del efecto de un muelle.

- **Doble Efecto**

  El aire a presión es el encargado de empujar al embolo en las dos direcciones, al poder ser introducido de formas arbitrarias en cualquiera de las dos cámaras.

b) Motores Neumáticos

Se consigue el movimiento de rotación de un eje mediante el aire a presión.

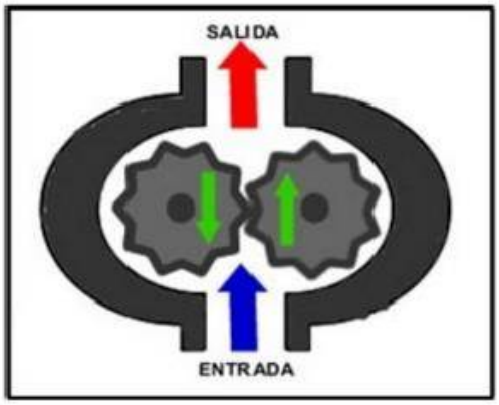


Figura 3.34. Motor Neumático

Se dividen en:

- **Motores de Aletas Rotativas**

  Al entrar aire a presión en uno de los compartimientos, formados por dos aletas, la carcasa tiende a girar hacia un sentido.

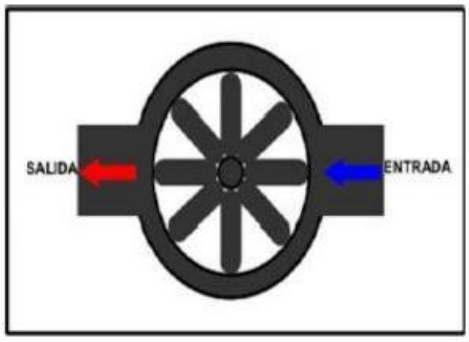


Figura 3.35. Motor de Aleta Rotativa

- **Motores de Pistones Axiales**

  Tienen un eje de giro solidario a un tambor que se obliga a girar por las fuerzas que ejercen varios cilindros, que se apoyan sobre un plano inclinado.

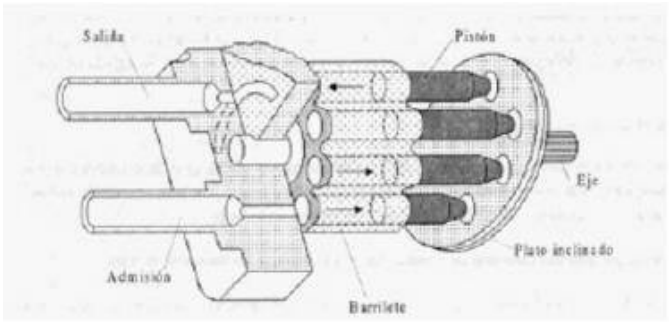


Figura 3.36. Motor de Pistón Axial

3.6.1.2. Actuadores Hidráulicos

En este tipo de actuador su fuente de energía son Aceites Minerales a una presión comprendida entre los 50 y 100 bar, llegando a superar los 300 bar.

Figura 3.37. Actuador Hidráulico

Características:

- La precisión obtenida es mayor que la de los actuadores neumáticos.
- Permiten desarrollar elevadas fuerzas y pares.
- Estabilidad frente a cargas estáticas.
- Elevada capacidad de carga y relación potencia peso.
- Autolubricación y robustez

Se dividen en:

a) **Tipo Cilindro**

b) **Motores de Aletas y Pistones**

3.6.1.3. Actuadores Eléctricos

Este tipo de actuador presenta gran control, sencillez y precisión, por tanto son los más utilizados en la actualidad.

Figura 3.38. Actuador Eléctrico

Existen tres tipos:

a) Motores de Corriente Continua DC

Un motor eléctrico de Corriente Continua es esencialmente una máquina que convierte energía eléctrica en movimiento o trabajo mecánico, a través de medios electromagnéticos.

FUNDAMENTOS DE OPERACIÓN DE LOS MOTORES ELÉCTRICOS

En magnetismo se conoce la existencia de dos polos: polo norte (N) y polo sur (S), que son las regiones donde se concentran las líneas de fuerza de un imán. Un motor para funcionar se vale de las fuerzas de atracción y repulsión que existen entre los polos. De acuerdo con esto, todo motor tiene que estar formado con polos alternados entre el estator y el rotor, ya que los polos magnéticos iguales se repelen, y polos magnéticos diferentes se atraen, produciendo así el movimiento de rotación.

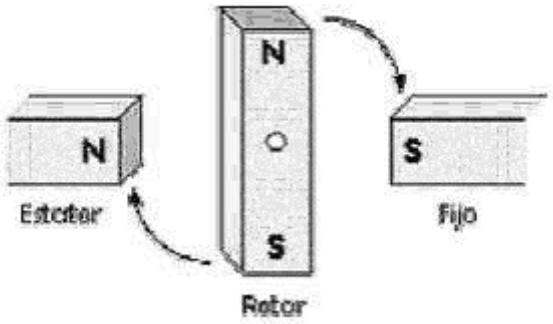


Figura 3.39. Funcionamiento del Motor de Corriente Continua

Un motor eléctrico opera primordialmente en base a dos principios: El de inducción, descubierto por Michael Faraday en 1831; que señala, que si un conductor se mueve a través de un campo magnético o está situado en las proximidades de otro conductor por el que circula una corriente de intensidad variable, se induce una corriente eléctrica en el primer conductor. Y el principio que André Ampere observo en 1820, en el que establece: que si una corriente pasa a través de un conductor situado en el interior de un campo magnético, éste ejerce una fuerza mecánica o f.e.m. (fuerza electromotriz), sobre el conductor.

El movimiento giratorio de los motores de C.C. se basa en el empuje derivado de la repulsión y atracción entre polos magnéticos. Creando campos constantes convenientemente orientados en estator y rotor, se origina un par de fuerzas que obliga a que la armadura (también le

llamamos así al rotor) gire buscando "como loca" la posición de equilibrio.

b) Motores paso a paso

En estos motores, la señal de control son trenes de impulsos que van actuando rotativamente sobre una serie de electroimanes dispuestos en el extractor, por cada pulso recibido, el rotor del motor gira un determinado número discreto de grados.

Para conseguir el giro de un motor en un determinado número de grados, las bobinas del estator deben ser excitadas a una frecuencia que determina la velocidad de giro.

Figura 3.40. Motor paso a paso

Se divide en:

- **Imanes permanentes**

  El rotor posee una polarización magnética constante, gira para orientar sus polos de acuerdo al campo magnético creado por las fases del estator.

- **Reluctancia variable**

  El rotor está formado por un material ferromagnético que tiende a orientarse de modo que facilite el camino de las líneas de fuerza del campo magnético generada por las bobinas del estator.

- **Híbridos**

  Se caracteriza por tener varios dientes en el estator y en el rotor, el rotor con un imán concéntrico magnetizado axialmente alrededor de su eje. Se puede ver que esta configuración es una

mezcla de los tipos de reluctancia variable e imán permanente. Este tipo de motor tiene una alta precisión y alto par, se puede configurar para suministrar un paso angular tan pequeño como 1.8°.

c) Motores de Corriente Alterna

Las mejoras que se han introducido hacen que se presenten como un gran competidor de los motores DC, esto se debe principalmente a:

- Construcción de rotores síncronos sin escobillas.
- Uso de convertidores estáticos.
- Empleo de la microelectrónica.

Figura 3.41. Motor de Corriente Alterna

Se divide en:

- **Motores Síncronos**

  Es un tipo de motor de corriente alterna en el que la rotación del eje está sincronizada con la frecuencia de la corriente de alimentación; el período de rotación es exactamente igual a un número entero de ciclos de CA. Su velocidad de giro es constante y depende de la frecuencia de la tensión de la red eléctrica a la que esté conectado y por el número de pares de polos del motor, siendo conocida esa velocidad como "velocidad de sincronismo".

Este tipo de motor contiene electromagnetos en el estator del motor que crean un campo magnético que rota en el tiempo a esta velocidad de sincronismo.

- **Motores Asíncronos**

  Es un tipo de motor de corriente alterna en el que la corriente eléctrica del rotor necesaria para producir torsión es inducida por inducción electromagnética del campo magnético de la bobina del estator. Por lo tanto un motor de inducción no requiere una conmutación mecánica aparte de su misma excitación o para todo o parte de la energía transferida del estator al rotor, como en los universales, DC y motores grandes síncronos. El primer prototipo de motor eléctrico capaz de funcionar con corriente alterna fue desarrollado y construido por el ingeniero Nikola Tesla y presentado en el American Institute of Electrical Engineers (en español, Instituto Americano de Ingenieros Eléctricos, actualmente IEEE) en 1888.

  El motor asíncrono trifásico está formado por un rotor, que puede ser de dos tipos: a) de jaula de ardilla; b) bobinado, y un estator, en el que se encuentran las bobinas inductoras. Estas bobinas son trifásicas y están desfasadas entre sí 120º en el espacio. Según el Teorema de Ferraris, cuando por estas bobinas circula un sistema de corrientes trifásicas equilibradas, cuyo desfase en el tiempo es también de 120º, se induce un campo magnético giratorio que envuelve al rotor.

  Este campo magnético variable va a inducir una tensión en el rotor según la Ley de inducción de Faraday: La diferencia entre el motor a inducción y el motor universal es que en el motor a inducción el devanado del rotor no está conectado al circuito de excitación del motor sino que está eléctricamente aislado. Tiene barras de conducción en todo su largo, incrustadas en ranuras a distancias uniformes alrededor de la periferia. Las barras están conectadas con anillos (en cortocircuito como dicen los electricistas) a cada extremidad del rotor. Están soldadas a las extremidades de las barras.

  Este ensamblado se parece a las pequeñas jaulas rotativas para ejercitar a mascotas como hámster y por eso a veces se llama "jaula de ardillas", y los motores de inducción se llaman motores de jaula de ardilla.

CAPÍTULO 4

HARDWARE

4.1 DESCRIPCIÓN FISICA DEL DISEÑO DEL SISTEMA

El Sistema de la Mini Planta de Control de Nivel consiste en el control de nivel del Líquido del tanque por medio de un controlador diseñado con las herramientas de la Placa de Evaluación STM32-V3.

La planta el cual está constituido por varios componentes:

- Placa de Evaluación STM32-V3
- Electroválvula
- Sensor Ultrasónico

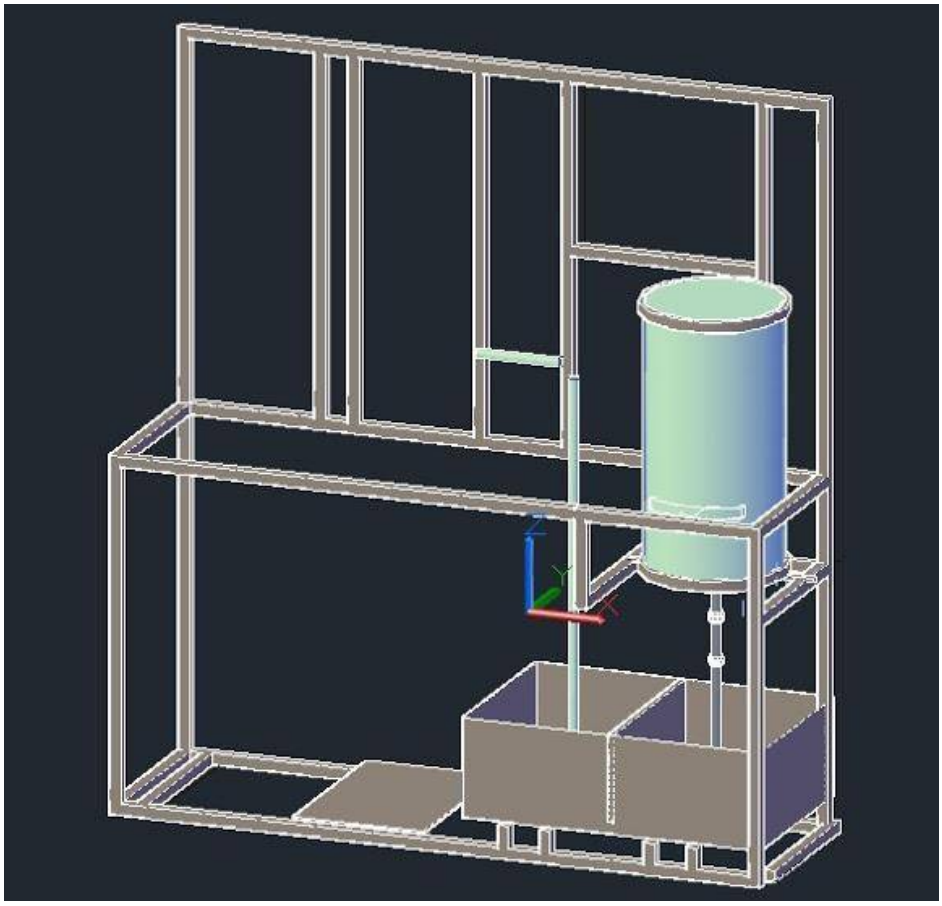


Figura 4.1. Mini Planta de Control de Nivel

4.1.1. PLACA DE EVALUACION MINI STM32-V3

En el mercado de los microcontroladores existen infinidad de placas usando microcontroladores de diferentes marcas, pero para este proyecto de tesis se decidió usar la placa MINI STM32-V3 pues no solo ofrece todas las características descritas anteriormente si no que es una plataforma compacta y versátil que usa un microcontrolador ARM Cortex M3 muy potente que también es de bajo costo y alto rendimiento accesible a cualquier aficionado a la electrónica.

4.1.1.1. Historia

El diseño del ARM comenzó en 1983 como un proyecto de desarrollo en la empresa Acorn Computers. Sophie Wilson y Steve Furber lideraban el equipo, cuya meta era, originalmente, el desarrollo de un procesador avanzado, pero con una arquitectura similar a la del MOS 6502.

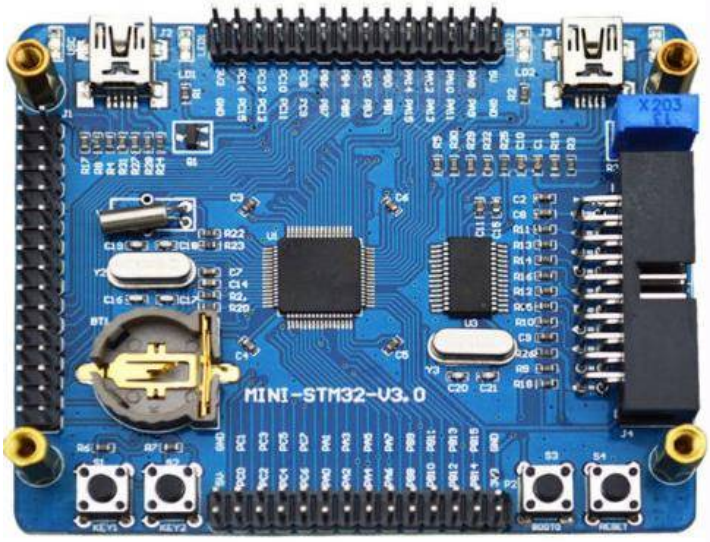


Figura 4.2. MINI STM32-V3

El equipo termino el diseño preliminar y los primeros prototipos del procesador en el año 1985, al que llamaron ARM1. La primera versión utilizada comercialmente se bautizó como ARM2 y se lanzó en el año 1986.

La arquitectura del ARM2 posee un bus de datos de 32 bits y ofrece un espacio de direcciones de 26 bits, junto con 16 registros de 32 bits. Uno de estos registros se utiliza como contador de programa, aprovechándose sus 4 bits superiores y los 2 inferiores para contener los flags de estado del procesador.

El ARM2 es probablemente el procesador de 32 bits útil más simple del mundo, ya que posee solo 30.000 transistores. Su simplicidad se debe a que no está basado en micro código (sistema que suele ocupar en torno a la cuarta parte de la cantidad total de transistores usados en un procesador) y a que, como era común en aquella época, no incluye caché.

Gracias a esto, su consumo en energía es bastante bajo, a la vez que ofrece un mejor rendimiento que un 286. Su sucesor, el ARM3, incluye una pequeña memoria caché de 4 KB, lo que mejora los accesos a memoria repetitivos.

A finales de los años 80, Apple Computer comenzó a trabajar con Acorn en nuevas versiones del núcleo ARM. En Acorn se dieron cuenta de que el hecho de que el fabricante de un procesador fuese también un fabricante de ordenadores podría echar para atrás a los clientes, por lo que se decidió crear una nueva compañía llamada Advanced RISC Machines, que sería la encargada del diseño y gestión de las nuevas generaciones de procesadores ARM. Ocurría esto en el año 1990.

Este trabajo derivó en el ARM6, presentado en 1991. Apple utilizó el ARM 610 (basado en el ARM6), como procesador básico para su innovador PDA, el Apple Newton. Por su parte, Acorn lo utilizó en 1994 como procesador principal en su RiscPC.

El núcleo mantuvo su simplicidad a pesar de los cambios: en efecto, el ARM2 tiene 30.000 transistores, mientras que el ARM6 sólo cuenta con 35.000. La idea era que el usuario final combinara el núcleo del ARM con un número opcional de periféricos integrados y otros elementos, pudiendo crear un procesador completo a la medida de sus necesidades.

La mayor utilización de la tecnología ARM se alcanzó con el procesador ARM7TDMI, con millones de unidades en teléfonos móviles y sistemas de videojuegos portátiles.

DEC licenció el diseño, lo cual generó algo de confusión debido a que ya producía el DEC Alpha, y creó el StrongARM. Con una velocidad de reloj de 233 MHz, este procesador consumía solo 1 W de potencia (este consumo de energía se ha reducido en versiones más recientes). Esta tecnología pasó posteriormente a manos de Intel, como fruto de un acuerdo jurídico, que la integró en su línea de procesadores Intel i960 e hizo más ardua la competencia.

Freescale (una empresa que derivó de Motorola en el año 2004), IBM, Infineon Technologies, OKI, Texas Instruments, Nintendo, Philips, VLSI, Atmel, Sharp, Samsung y STMicroelectronics también licenciaron el diseño básico del ARM. El diseño del ARM se ha convertido en uno de los más usados del mundo, desde discos duros hasta juguetes. Hoy en día, cerca del 75 % de los procesadores de 32 bits poseen este chip en su núcleo.

4.1.1.2. El Core Cortex-M3

El núcleo Cortex-M3 presenta un proceso de interrupción rápido y de alta determinación, por lo que es muy apropiado para aplicaciones de microcontroladores. Además Cortex-M3 pone en práctica la nueva arquitectura Thumb-2 de conjunto de instrucciones mixtas.

El núcleo central del Cortex-M3 está basado en la arquitectura Harvard caracterizada por buses separados para instrucciones y datos como lo muestra la figura 3.2. El core CortexM3 puede desarrollar varias operaciones en paralelo. Este núcleo está segmentado en tres etapas Fetch (búsqueda de instrucción), decodificación de instrucción y Ejecución.

El core Cortex-M3 contiene una avanzada ALU la cual realiza operaciones de división y multiplicación de 32 bits por hardware, lógica de control e interfaces con otros componentes del procesador.

Como se mencionó, el Cortex-M3 es un procesador de 32-bit, con un ancho de banda para el flujo de datos de 32 bits, así como banco de registros e interface de memoria. Tiene 13 registros de propósito general, dos punteros de pila, un registro de búsqueda, un contador de programa y registros especiales, incluido un registro de estado.

Dicho procesador cuenta con un sistema de mapa de memoria sencillo, es un mapa de memoria fijo de 4 gigabytes, dividiendo el espacio de memoria con espacios predefinidos y dedicados para código (espacio de código), SRAM (espacio de memoria), memoria/dispositivos externos y periféricos externos/internos.

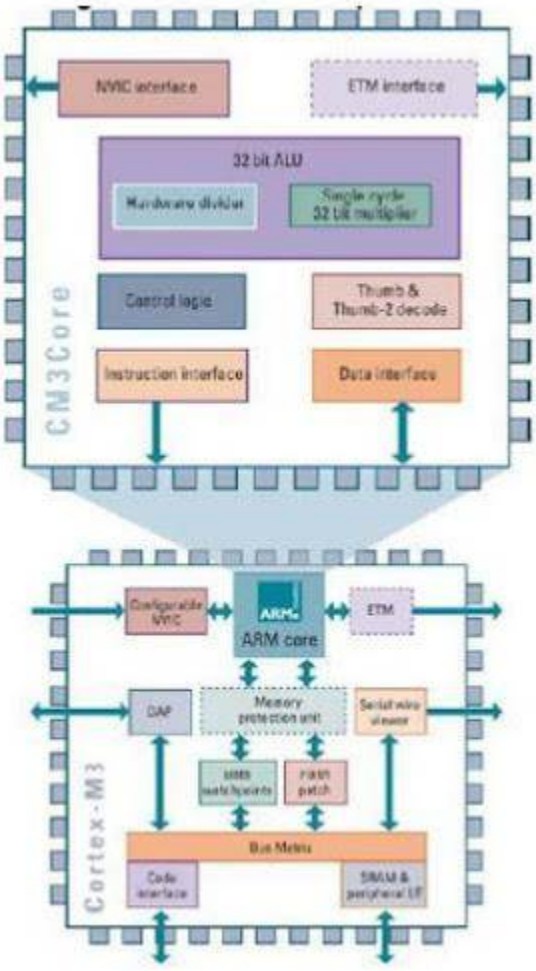


Figura 4.3. Arquitectura Arm

4.1.1.3. Características Técnicas

La placa de evaluación MINI STM32-V3 ofrece múltiples posibilidades para aplicaciones relacionadas con la electrónica y la automática en general. En la figura se puede observar una imagen real de la placa de evaluación empleada, con todos los periféricos que la componen. Las características técnicas más importantes de la placa se citan a continuación:

- MCU: STM32F103RBT6 ARM CORTEX M3 de 32 bits con 128 K Bytes programa
- Flash, 20K Bytes de RAM, USB, CAN, I2C x2, x2 de ADC de 12 bits, UART x3, x2 SPI, temporizadores x3, hasta la operación 72Mhz.
- Conector JTAG estándar con distribución de pines ARM 2x10 para la programación / depuración con ARM-JTAG.
- Conector USB.
- Botón de usuario.
- Botón RESET.
- LED de estado.
- Fuente de alimentación del LED
- El regulador de voltaje de 3.3V a bordo con un máximo de 800 mA de corriente.
- Sola fuente de alimentación: toma energía del puerto USB o la extensión de pines del conector.
- 8 Mhz oscilador de cristal.
- 32768 Hz cristal y conector de la batería de respaldo de RTC.
- Cabeceras de extensión para todos los puertos uC.
- PCB: FR-4, 1,5 mm (0,062”), mascara de soldadura, el componente de impresión serigráfica.

En la figura se puede apreciar una imagen a modo de diagrama de bloques extraída del Manual de Usuario de la placa de evaluación con el microcontrolador y los periféricos que componen.

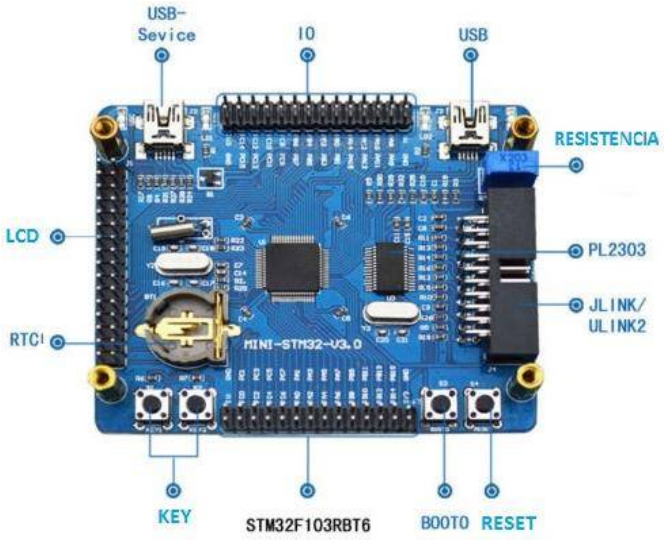


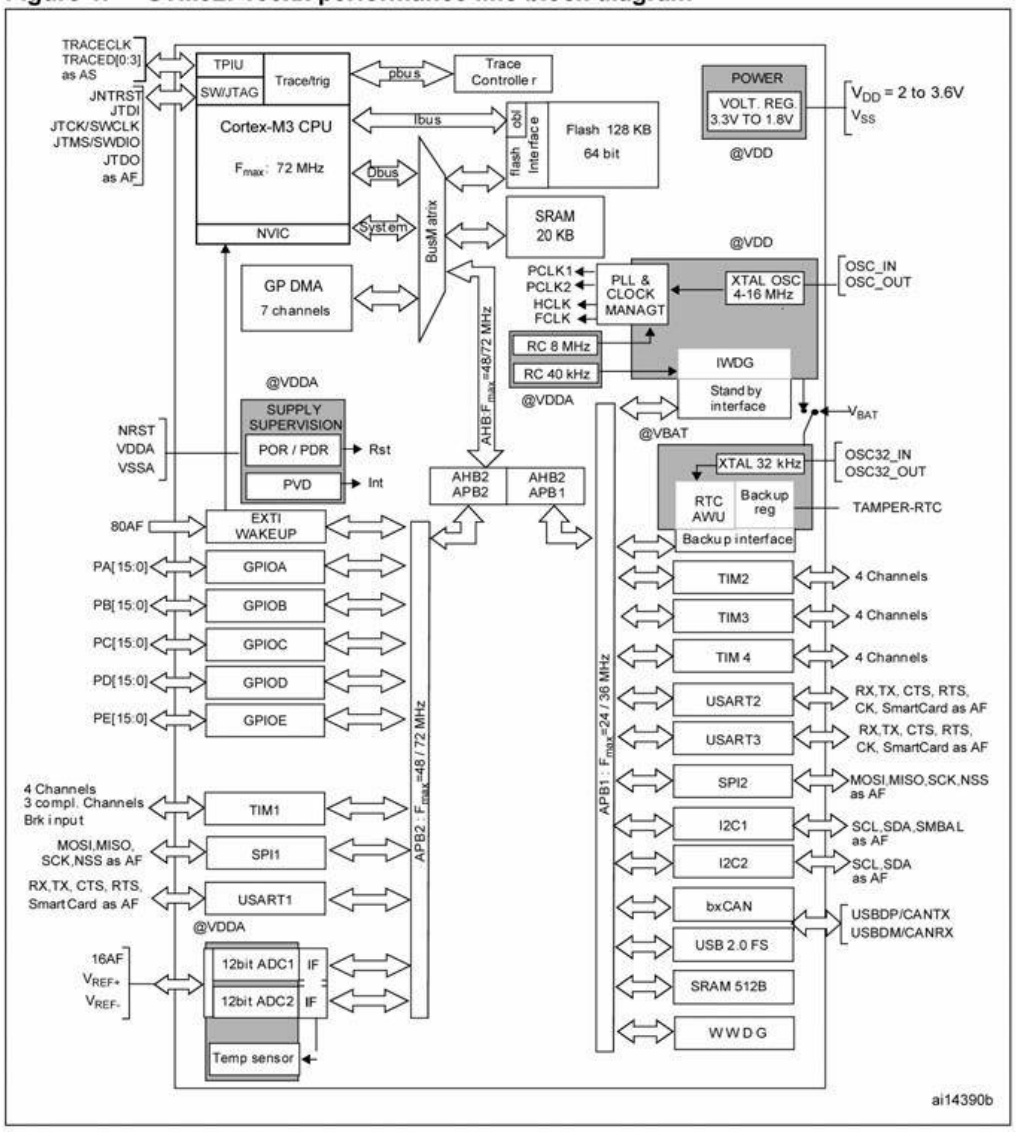
Figura 4.4. Layout

4.1.1.4. Microcontrolador

La placa de evaluación MINI STM32-V3 utiliza un microcontrolador STM32F103RBT6 ARM Cortex M3 de la familia ST Microelectronics Inc basado en un procesador (Véase figura 4.7) con las siguientes características:

- Reloj de la CPU hasta 72Mhz.
- 128KB FLASH.
- 20KB RAM.
- Canales x7 DMA.
- RTC
- WDT
- Timers x3 +1.
- X2 SPI.
- X2 I2C.
- USART x3.
- X1 USB.
- X1 CAN (multiplexado con USB, por lo tanto no se puede utilizar en el mismo tiempo).
- GPIO hasta 51 (multiplexado con periféricos).
- X2 ADC de 12 bits.
- Tensión de funcionamiento 2.0-3.6V.
- Temperatura de-40C +85 C.

Figure 1. STM32F103xx performance line block diagram



1. T_A = –40 °C to +105 °C (junction temperature up to 125 °C).
2. AF = alternate function on I/O port pin.

Figura 4.5. Diagrama de Bloques

Figura 4.6. Microcontrolador

4.1.1.5. Mapa de Memoria

El mapa de memoria se muestra en la figura 4.8 con los principales registros a configurar.

4.1.16. Esquema Eléctrico MINI STM32-V3

La siguiente figura 4.9 muestra la distribución y conexión de los diferentes periféricos que posee la placa de evaluación MINI STM32-V3.

4.1.1.7. Cmsis

CMSIS, siglas en ingles de Cortex Microcontroller Software Interface Standard, al cual le otorgamos gran parte del éxito que está teniendo la arquitectura en los últimos tiempos.

Este interfaz estándar de software suele estar integrado en las librerías que proporcionan los fabricantes y permite el acceso a una serie de funciones del sistema y periféricos transversales a todos los modelos. Nació en 2008 con el propósito de mejorar la portabilidad y reusabilidad de código creado, evitar problemas de compatibilidad de drivers, facilitar el trabajo a los creadores de herramientas de desarrollo y compilación y, en definitiva, crear una plataforma que permita un desarrollo más estándar, rápido y sencillo.

En cualquier caso, gran parte de los aspectos relacionados con la arquitectura quedan enmascarados cuando este tipo de microcontroladores se programan en lenguaje C. La mayoría de los fabricantes de microcontroladores proporcionan librerías escritas en C que permiten controlar todos los aspectos del dispositivo. Dado que los compiladores modernos generan código muy eficiente, cada vez tiene menos sentido utilizar lenguaje ensamblador.

CMSIS define:

- Una manera común de acceso a registros de periféricos del Core y de definir vectores de excepción.

- Los nombres de los registros de los periféricos del Core y de los vectores de excepción del mismo.

- Una interfaz independiente del dispositivo para RTOS Kernels incluyendo un canal de depuración.

- Mediante el uso de software CMSIS compatible, el usuario puede fácilmente reutilizar el código. CMSIS tiene por objeto permitir la combinación de componentes de software de múltiples proveedores de componentes.

Memory mapping

The memory map is shown in *Figure 7*.

Figure 7. Memory map

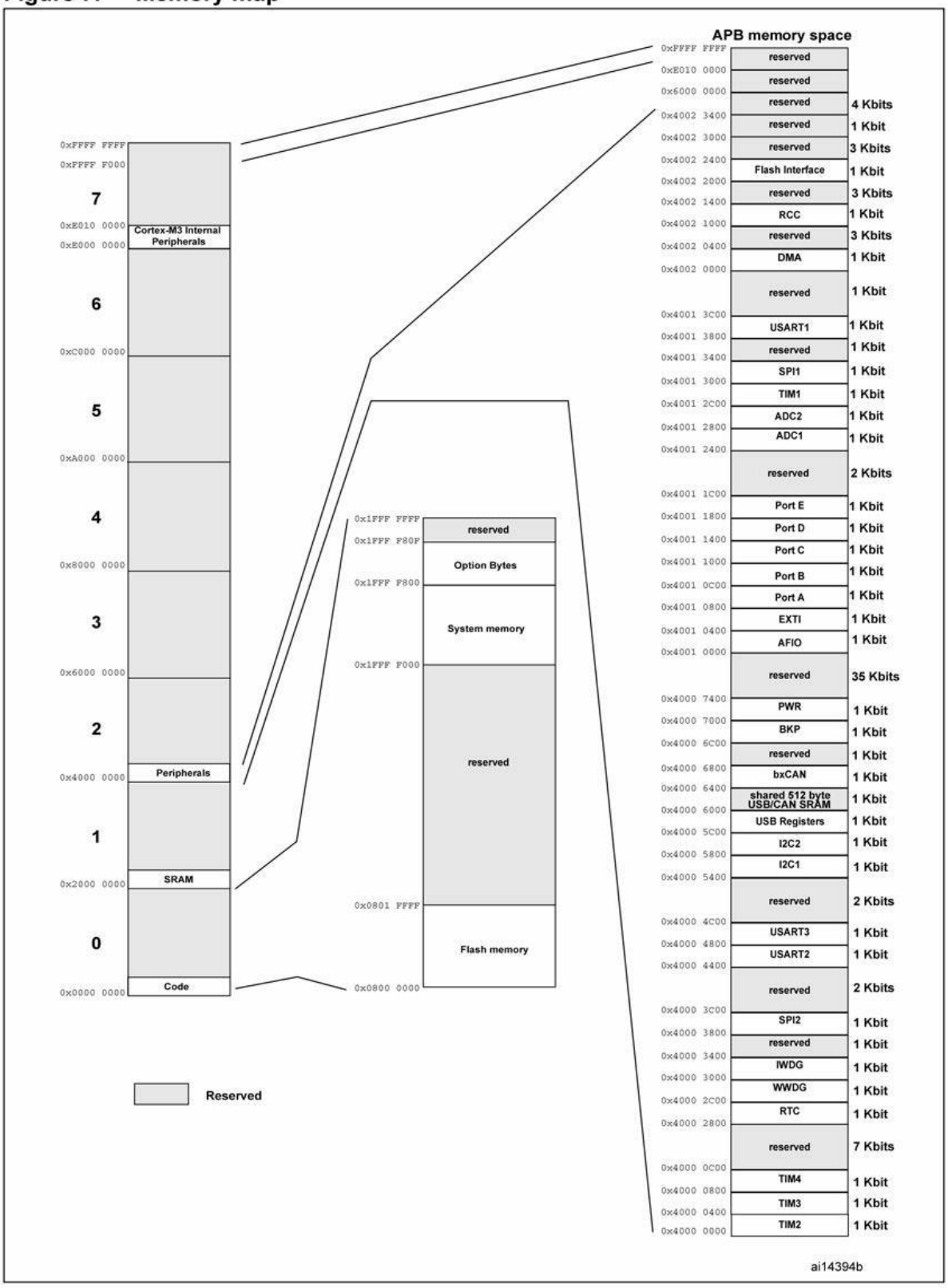


Figura 4.7. Mapa de Memoria

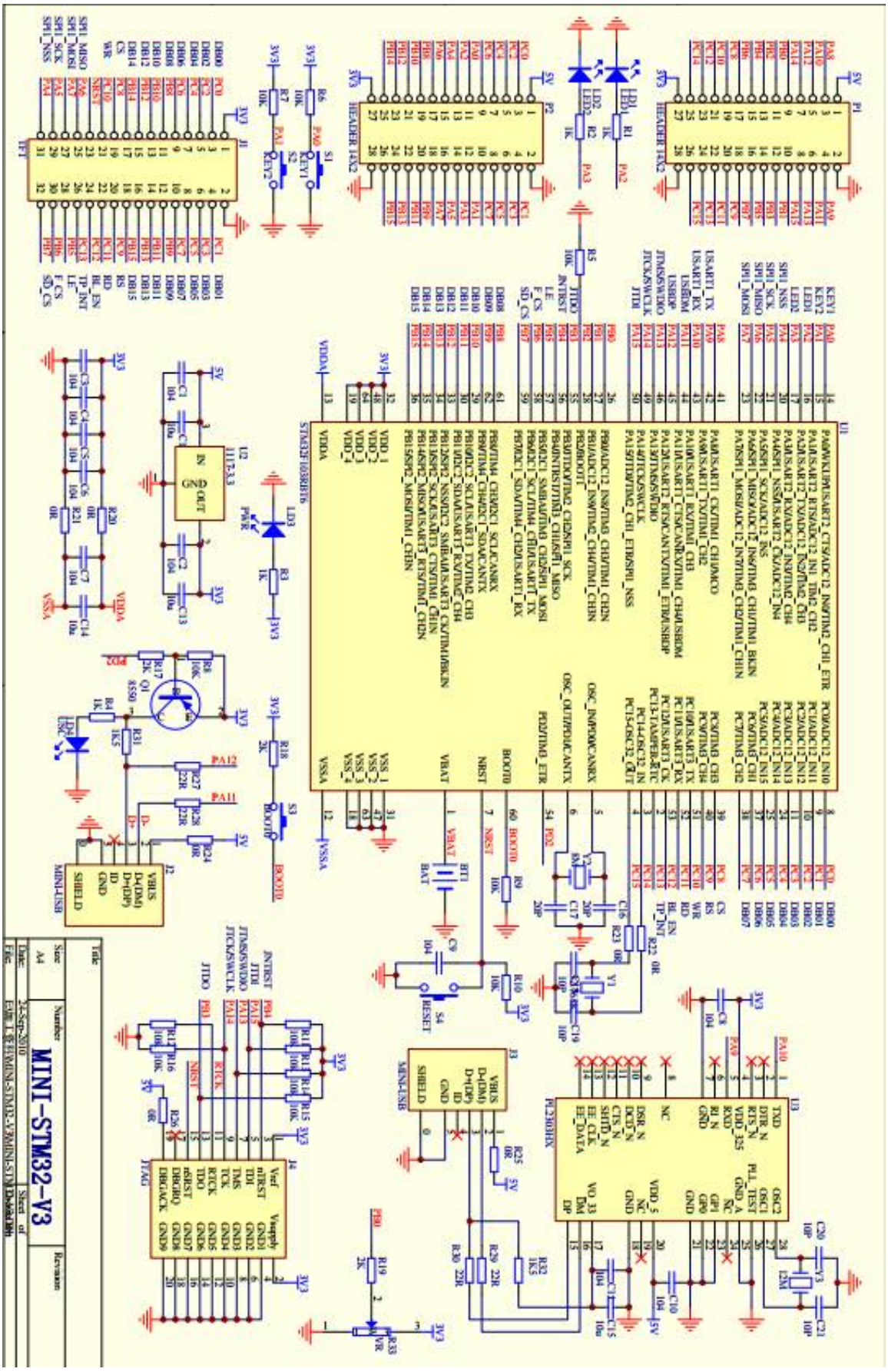


Figura 4.8. Esquema Eléctrico MINI STM32-V3

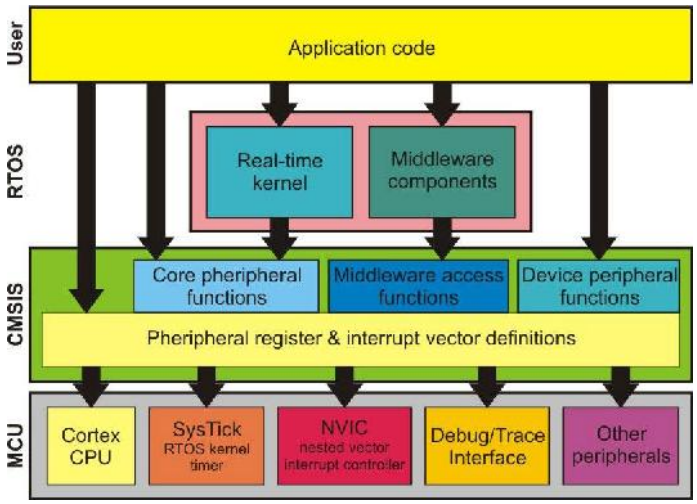


Figura 4.9. CMSIS

4.1.1.8. Librerías

STMicroelectronics facilita una librería descargable desde su página web llamada STM32L1xx Standard Peripherals library y STM32F10xxx USB Library. Ésta cubre dos niveles de abstracción:

- Un completo mapeo de registros con todos los bits, campos de bits y registros declarados en lenguaje C. Esto evita conocer a fondo cada periférico ya que son de elevada complejidad y con esto se facilita el aprendizaje en fases iniciales.

- Colección de funciones y estructuras de datos que cubren todas las funcionalidades de los periféricos.

Cada driver consiste en un conjunto de funciones cubriendo todas las funcionalidades de periféricos. El desarrollo de cada driver está basado en un API común (Application Programming Interface) que estandariza la estructura del driver.

El código de los drivers ha sido escrito en C. Está documentado y es conforme a los requisitos MISRA-C 2004 y CMSIS. La librería es independiente del toolchain empleado, solo los ficheros de inicio startup dependen del toolchain.

La ventaja de utilizar esta librería reside en el tiempo que se necesita para comenzar a desarrollar aplicaciones para STM32L1xx. Gracias a ella, no necesitamos conocer en profundidad todas las facetas y posibilidades del periférico que deseamos programar. Sin embargo, tiene la desventaja que no optimiza el tamaño de la aplicación ni la velocidad de ejecución. En casos en los que tengamos restricciones de tamaño o tiempo, es conveniente utilizar la librería como una referencia para aprender como programar el periférico.

A) STM32L1xx Standard Peripherals library: Podemos descargar de la página web de STMicroelectronics la librería en un fichero comprimido zip. La extracción de este fichero generará la carpeta: STM32F10x StdPeriph Lib V3.5.0. El contenido de esta carpeta es el que se muestra en la (figura 4.11).

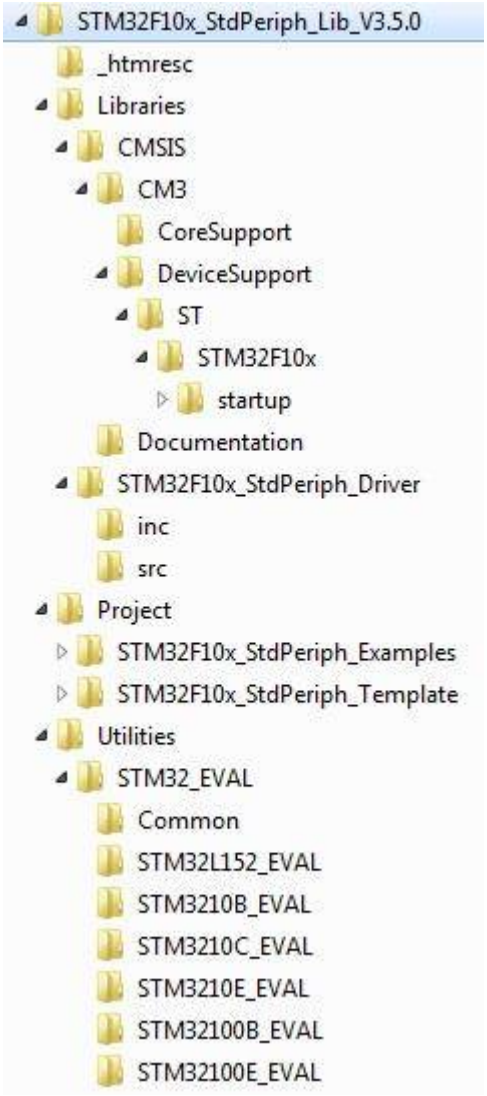


Figura 4.10. STM32L1xx Standard Peripherals Library

- _htmresc: Esta carpeta contiene todas las páginas de recursos de html de la librería.

- Libraries: Contiene todos los ficheros CMSIS y STM32L1xx Standard Peripheral's Drivers.

    - CMSIS: Contiene todos los ficheros CMSIS de STM32L1xxxx: device peripheral access layer y core peripheral access layer.

- STM32L1xx_StdPeriph_Driver: Contiene todos los subdirectorios y ficheros que constituyen el núcleo de la librería:

    - inc: Contiene los ficheros de cabecera de los drivers de periféricos. Contiene un fichero cabecera por cada driver de periférico.

    - src: Contiene los ficheros fuente de los drivers de periféricos. Contiene un fichero fuente por cada driver de periférico.

- Project: Esta carpeta contiene los template para diferentes toolchain y ejemplos de cada periférico.

    - STM32L1xx_StdPeriph_Examples: Esta carpeta contiene una subcarpeta por cada periférico, y dentro de ella, ejemplos de código para aprender a usar dicho periférico.

    - STM32L1xx_StdPeriph_Templates: Esta carpeta contiene templates para proyectos para los toolchain: EWARM, MDK-ARM, RIDE, HiTOP, y TrueSTUDIO.

- Utilities:

    - STM32 EVAL: Implementa una capa de abstracción para que el usuario interactue con la placa: botones, LEDs, LCD, puertos COM, etc. Contiene varias subcarpetas en función de la placa de evaluación que estemos utilizando.

4.1.2. SENSOR

Es un dispositivo que, a partir de la energía del medio donde se mide, da una señal de salida transductible que es función de la variable de medida. La ampliación de los sentidos para adquirir un conocimiento de cantidades físicas que por su naturaleza o tamaño, no pueden ser percibidas directamente por los sentidos.

4.1.2.1. Sensor de Ultrasonido

En la figura 4.12 se muestra un sensor ultrasónico. Los ultrasonidos son una radiación mecánica de frecuencia superior a los audibles (20Khz). Toda radiación al incidir sobre un objeto, en parte se refleja, en parte se transmite y en parte es absorbida. Si además hay un movimiento relativo entre la fuente de radiación y el reflector, se produce un cambio de frecuencia de la radiación (efecto Doppler).Todas estas

propiedades de la interacción de una radiación con un objeto han sido aplicadas en mayor o menor grado a la medida de diversas magnitudes físicas. El poder de penetración de la radiación permite que muchas de estas aplicaciones sean totalmente no invasivas, es decir, que no acceda al interior del recinto donde se producen los cambios que se desean detectar.

En función del tiempo que tarda el sonido en rebotar y volver, se calcula la distancia a la que se encuentra dicho objeto.

Figura 4.11. Sensor Ultrasónico

4.1.2.2. Funcionamiento del sensor ultrasónico

El ultrasonido es sonido exactamente igual al que escucha el ser humano normalmente, pero con una frecuencia mayor a la máxima audible por el oído humano. Ésta comienza desde unos 16 Hz y tiene un límite superior de aproximadamente 20 KHz, mientras que se va a utilizar sonido con una frecuencia de 40 KHz. A este tipo de sonidos es a lo que se denomina Ultrasonidos.

El funcionamiento básico de los sensores ultrasónicos como medidores de distancia se muestra en la figura 4.12, donde se tiene un receptor que emite un pulso de ultrasonido, el cual rebota sobre un determinado objeto y la reflexión de ese pulso es detectada por un receptor de ultrasonidos.

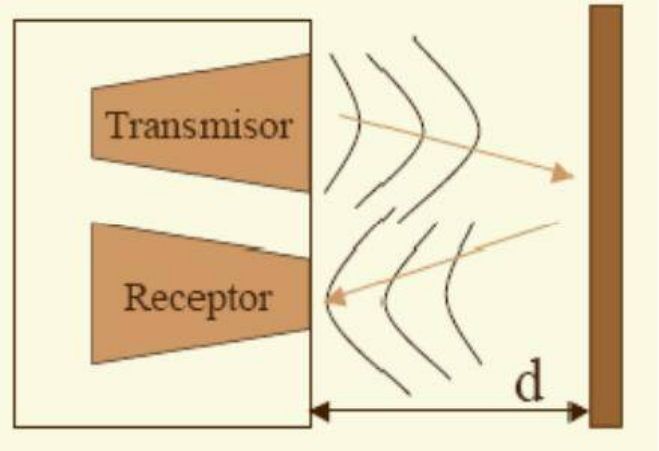


Figura 4.12. Principio de Funcionamiento de los Sensores Ultrasónicos

La mayoría de los sensores de ultrasonido de bajo costo se basan en la emisión de un pulso de ultrasonido cuyo lóbulo, o campo de acción, es de forma cónica. Midiendo el tiempo que transcurre entre la emisión del sonido y la percepción del eco se puede establecer la distancia a la que se encuentra el obstáculo que ha producido la reflexión de la onda sonora, mediante la fórmula:

$$d = \frac{1}{2} V \, . \, t$$

Donde V es la velocidad del sonido en el aire y t es el tiempo transcurrido entre la emisión y recepción del pulso.
A pesar de que su funcionamiento parece muy sencillo, existen factores inherentes tanto a los ultrasonidos como al mundo real, que influyen de una forma determinante en las medidas realizadas. Por tanto, es necesario un conocimiento de las diversas fuentes de incertidumbre que afectan a las medidas para poder tratarlas de forma adecuada, minimizando su efecto en el conocimiento del entorno que se desea adquirir. Entre los diversos factores que alteran las lecturas que se realizan con los sensores de ultrasonido cabe destacar

El campo de actuación del pulso que se emite desde un transductor de ultrasonido tiene forma cónica. El eco que se recibe como respuesta a la reflexión del sonido indica la presencia del objeto más cercano que se encuentra dentro del cono acústico y no especifica en ningún momento la localización angular del mismo, como se muestra en la figura 4.13. Aunque la opción de que el objeto detectado esté sobre el eje central del cono acústico, la posibilidad que el eco se haya producido por un objeto presente en la periferia del eje central no es en absoluto despreciable y ha de ser tenida en cuenta y tratada convenientemente.

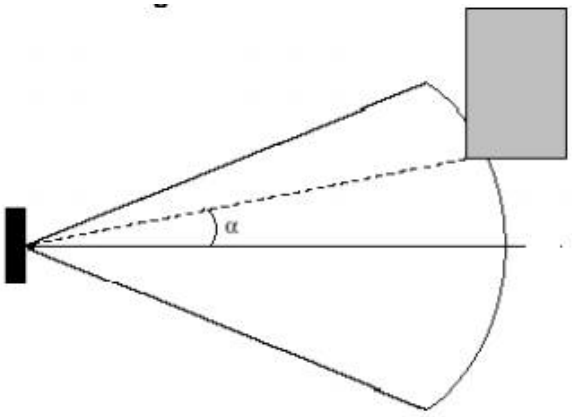


Figura 4.13. Incertidumbre angular en la medida de un Ultrasonido

La cantidad de energía acústica reflejada por el obstáculo depende en gran medida de la estructura de su superficie. Para obtener una reflexión altamente difusa del obstáculo, el tamaño de las irregularidades sobre la superficie reflectora debe ser comparable a la longitud de onda de la onda de ultrasonido incidente.

En los sensores de ultrasónicos de bajo coste se utiliza el mismo transductor como emisor y receptor. Tras la emisión del ultrasonido se espera un determinado tiempo a que las vibraciones en el sensor desaparezcan y esté preparado para recibir el eco producido por el obstáculo. Esto implica que existe una distancia mínima d (proporcional al tiempo de relajación del transductor) a partir de la cual el sensor mide con precisión. Por lo general, todos los objetos que se encuentren por debajo de esta distancia, d, serán interpretados por el sistema como que están a una distancia igual a la distancia mínima, como se muestra en la figura 4.14.

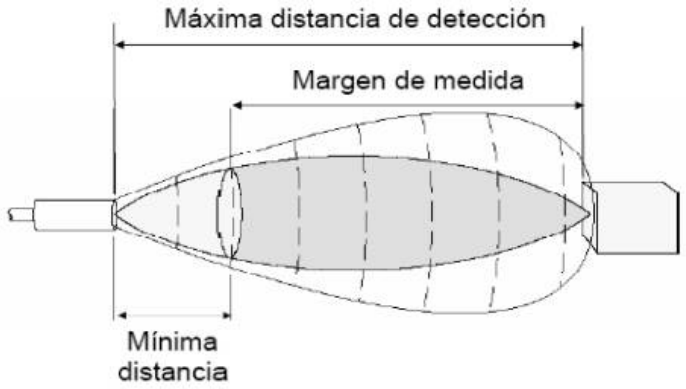


Figura 4.14. Márgenes de Detección de un sensor de Ultrasonido

Los factores ambientales tienen una gran repercusión sobre las medidas ya que las ondas de ultrasonido se mueven por un medio material que es el aire. La densidad del aire depende de la temperatura, influyendo este factor sobre la velocidad de propagación de la onda según la expresión:

$$V_S = V_{SO}\sqrt{1+\frac{T}{273}}$$

Siendo V_{so} la velocidad de propagación de la onda sonora a 0 ºC, y T la temperatura absoluta (grados Kelvin).

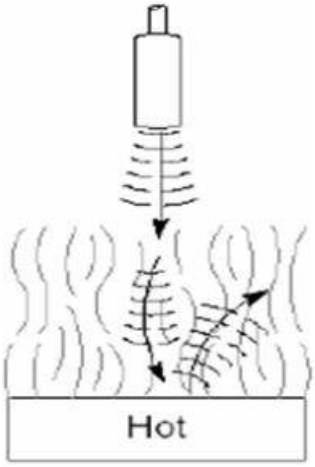


Figura 4.15. Perturbación de Señales de un Ultrasonido por la temperatura

Un factor de error muy común es el conocido como falsos ecos. Estos falsos ecos se pueden producir por razones diferentes: Puede darse el caso en que la onda emitida por el transductor se refleje varias veces en diversas superficies antes de que vuelva a incidir en el transductor (si es que incide). Este fenómeno, conocido como reflexiones múltiples, implica que la lectura del sensor evidencia la presencia de un obstáculo a una distancia proporcional al tiempo transcurrido en el viaje de la onda; es decir, una distancia mucho mayor que a la que está en realidad el obstáculo más cercano, que pudo producir la primera reflexión de la onda.

Otra fuente más común de falsos ecos, conocida como crosstalk, se produce cuando se emplea un cinturón de ultrasonidos donde una serie de sensores están trabajando al mismo tiempo. En este caso puede ocurrir (y ocurre con una frecuencia relativamente alta que un sensor emita un pulso y sea recibido por otro sensor que estuviese esperando el eco del pulso que él había enviado con anterioridad (o viceversa, lo cual se muestra en la figura 4.15.

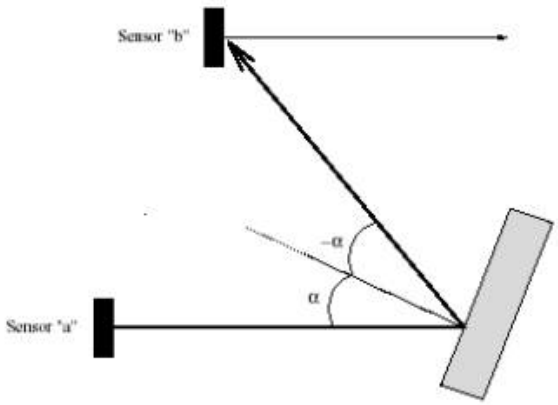


Figura 4.16. Reflexión de la Señal

Las ondas de ultrasonido obedecen a las leyes de reflexión de las ondas, por lo que una onda de ultrasonido tiene el mismo ángulo de incidencia y reflexión respecto a la normal a la superficie. Esto implica que si la orientación relativa de la superficie reflectora con respecto aleje del sensor de ultrasonido es mayor que un cierto umbral, el sensor nunca reciba el pulso de sonido que emitió.

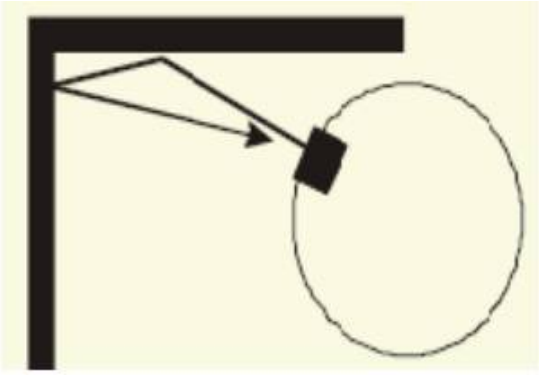


Figura 4.17. Reflexión angular de incidencia de un ultrasonido

4.1.2.3. SELECCIÓN DEL SENSOR

Los sensores de ultrasonido son los más económicos, fáciles de manipular y pueden detectar objetos en el orden de los metros sin necesidad de algún filtro o adaptación especial.

Después de investigar en los mercados diferentes tipos de sensores existentes, se optó por elegir el Modulo SR-04T Ultrasónico, por las siguientes características:

- Tiene un buen rendimiento y casi el mismo uso de HC-S04 módulo.
- Tamaño pequeño, fácil de usar.
- Baja tensión, bajo consumo de energía.
- Alta precisión
- Fuerte anti-jamming
- Integrado con alambre cerrado sonda resistente al agua, adecuado para mojado, ocasiones de medición duras

Especificaciones:

- Voltaje de funcionamiento: DC 5 V
- Corriente en reposo: 5mA
- Corriente de trabajo: 30mA
- Frecuencia de emisión acústica: 40 kHz
- Distancia más lejana: 4.5m
- Temperatura de trabajo: -10 – 70°C
- Temperatura de Almacenamiento: -20 – 80°C
- Angulo: menos de 50°
- Cableado: + 5 V (positivo); TRIGE (control), ECHO (recibir); GND (cátodo)

Figura 4.18. Modulo SR-04T Ultrasónico

4.1.2.4. Funcionamiento del Módulo Ultrasónico

Es un sensor de distancias por ultrasonidos desarrollado capaz de detectar objetos y calcular la distancia a la que se encuentra en un rango de 1,7 a 450 cm.

El sensor funciona por ultrasonidos y contiene toda la electrónica encargada de hacer la medición. Su uso es tan sencillo como enviar el pulso de arranque y medir la anchura del pulso de retorno. De muy pequeño tamaño, el sensor se destaca por su bajo consumo, gran precisión y bajo precio.

Funciona emitiendo impulsos de ultrasonidos inaudibles para el oído humano. Los impulsos emitidos viajan a la velocidad del sonido hasta alcanzar un objeto, entonces el sonido es reflejado y captado de nuevo por el receptor de ultrasonidos. Lo que hace el controlador incorporado es emitir una ráfaga de impulsos y a continuación empieza a contar el tiempo que tarda en llegar el eco. Este tiempo se traduce en un pulso de eco de anchura proporcional a la distancia a la que se encuentra el objeto. Registrando la duración del pulso es posible calcular la distancia en pulgadas/centímetros o en cualquier otra unidad de medida. Si no se detecta nada, entonces baja el nivel lógico de su línea de eco después de 30mS.

Proporciona un pulso de eco proporcional a la distancia. Si el ancho del pulso se mide en uS, el resultado se debe dividir entre 58 para saber el equivalente en centímetros, y entre 148 para saber el equivalente en pulgadas. US/58=cm o uS/148=pulgadas. Puede activarse cada 50mS, o 20 veces por segundo. Debería esperar 50ms antes de la siguiente activación, incluso si detecta un objeto cerca y el pulso del eco es más corto. De esta manera se asegura que el "bip" ultrasónico ha desaparecido completamente y no provocará un falso eco en la siguiente medición de distancia.

4.1.2.4.1. Modo 1, Compatibilidad con SRF04

Este modo emplea patillas separadas, una para aplicar el pulso de inicio o Trigger y otra para leer la anchura del pulso del ECO medido. Todos los

programas realizados para el SRF04 deben funcionar perfectamente en este modo, que se selecciona simplemente dejando la patilla “Mode” sin conectar (igual que en el SRF04).

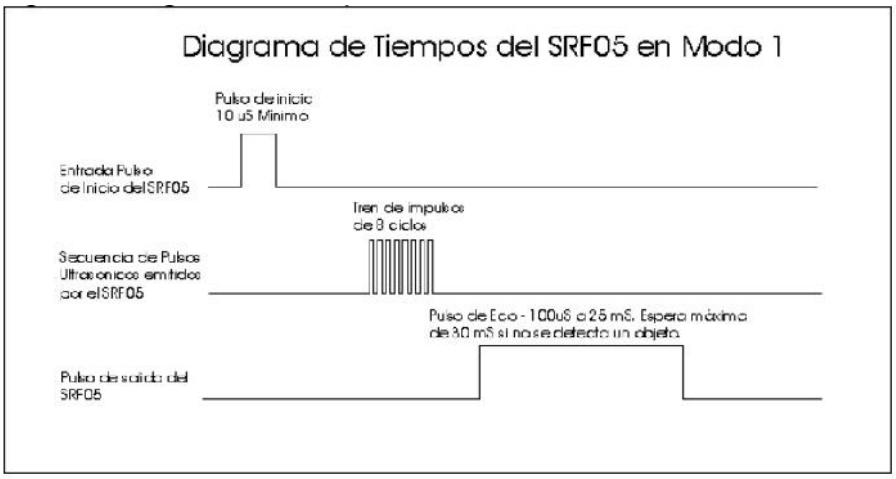


Figura 4.19. Diagrama de Tiempos en Modo 1

4.1.2.4.2. Modo 2, Patilla única para trigger y eco

Este modo permite emplear una única patilla para generar la señal de disparo o trigger y también para realizar la medida de la anchura del pulso de salida del ECO, lo que ahorra patillas en el microcontrolador central. Para emplear este modo basta con conectar la patilla “Modo” con GND. La señal de ECO aparecerá entonces en la misma patilla por la que se aplicó la señal de trigger. Esa patilla se debe configurar primero como salida para generar el disparo y luego como entrada para leer la duración del ECO. La sentencia PULSIN de los controladores más populares realiza esta reconfiguración de forma automática.

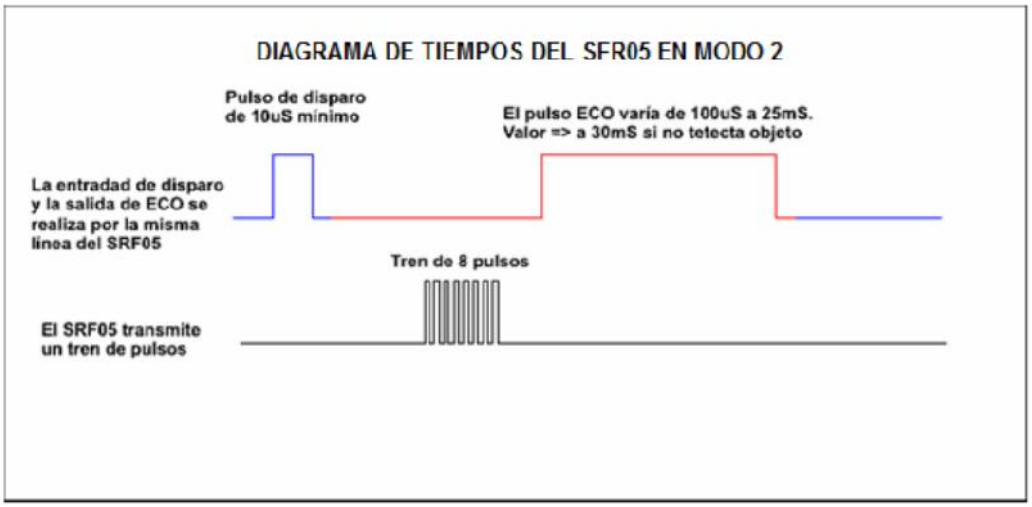


Figura 4.20. Diagrama de Tiempos en Modo 2

4.1.3. ELECTROVALVULA

Una electroválvula es una válvula electromecánica, diseñada para controlar el paso de un fluido por un conducto o tubería. La válvula se mueve mediante una bobina solenoide. Generalmente no tiene más que dos posiciones: abierto y cerrado, o todo y nada. Las electroválvulas se usan en multitud de aplicaciones para controlar el flujo de todo tipo de fluidos.

No se debe confundir la electroválvula con válvulas motorizadas, en las que un motor acciona el mecanismo de la válvula, y permiten otras posiciones intermedias entre todo y nada.

Figura 4.21. Electroválvula

Una electroválvula está compuesta por dos partes:

a) Una cabeza magnética constituida principalmente por una bobina, tubo, culata, anillo de desfasado, resorte(s).

b) Un cuerpo, con orificios de racordaje, obturados por clapet, membrana, pistón, etc. según el tipo de tecnología empleada.

La apertura y el cierre de la electroválvula están unida a la posición del núcleo móvil que se desplaza bajo el efecto del campo magnético provocado por la puesta con tensión de la bobina.

4.1.3.1. Tipos de Electroválvulas

a) Acción Directa

En esta familia de válvulas el flujo electromagnético actúa directamente en el émbolo que cierra o abre el orificio permitiendo que el líquido pase o pare (presión mínima requerida = 0 bar).

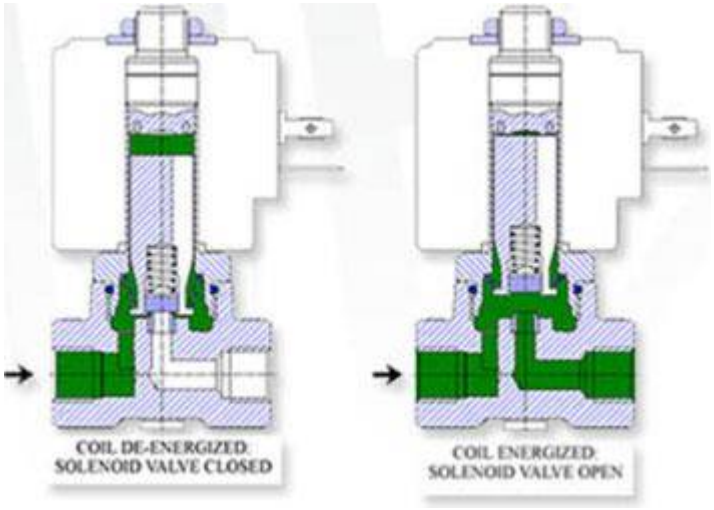


Figura 4.22. Acción Directa

b) Acción Indirecta

El orificio principal es abierto por el desequilibrio entre las presiones en las superficies del diafragma superior e inferior (o del pistón). Cuando se energiza la bobina el movimiento del émbolo causa la apertura del orificio de piloto y descarga el compartimiento superior del diafragma: el desequilibrio de la presión mueve el diafragma que abre el orificio principal (la presión mínima requerida es de 0.2 bar).

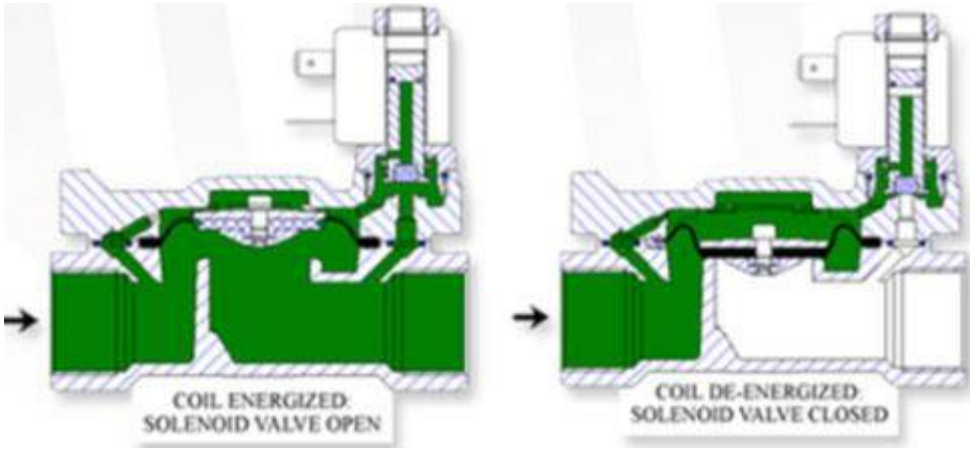


Figura 4.23. Acción Indirecta

c) Acción Mixta

En esta familia de válvulas la abertura del orificio principal es efectuada por el desequilibrio de presiones entre el cuerpo superior y el inferior combinando con la acción directa del émbolo que está fijo al diafragma mediante un resorte (presión mínima requerida = 0 barras).

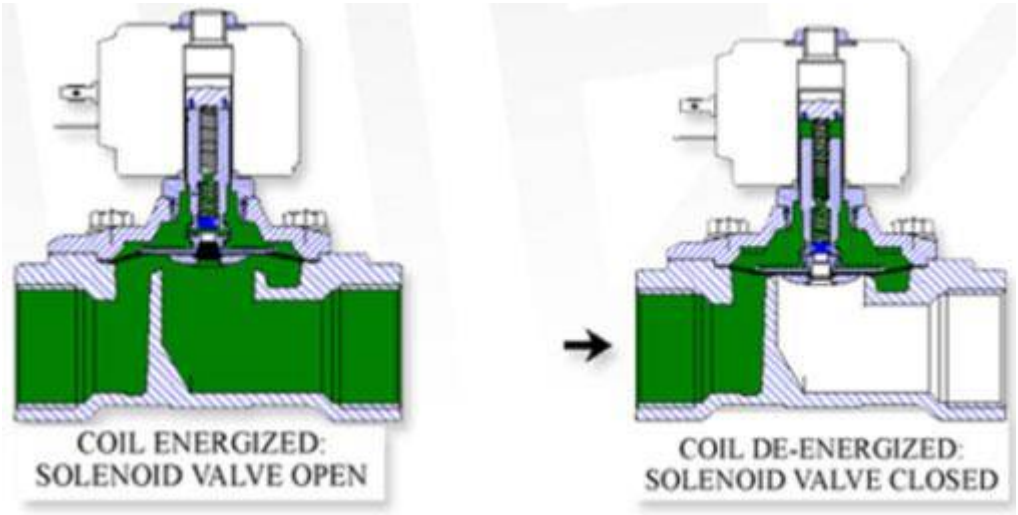


Figura 4.24. Acción Mixta

4.1.3.2. Número de Vías en las Electroválvulas

a) Las válvulas de 2 vías, son las válvulas más conocidas ya que tienen una entrada y una salida.

b) Las válvulas de 3 vías tienen una entrada, una salida y un escape, tal como se muestra en la siguiente imagen:

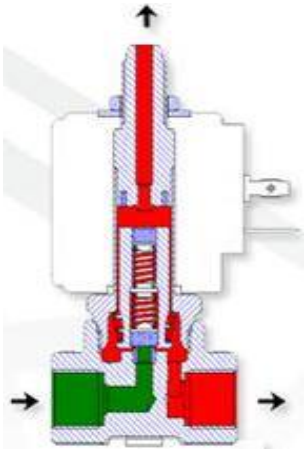


Figura 4.25. Número de Vías en las Electroválvulas

4.1.3.3. Normalmente Cerrada o Normalmente Abierta

Las válvulas de acción directa e indirecta pueden ser normalmente cerrada (NC) o normalmente abierta (NA).

Las válvulas normalmente cerradas, no dejan pasar el fluido cuando están en reposo y cuando son energizadas se abren dejando pasar el fluido. Por otro lado las válvulas normalmente abiertas dejan pasar el fluido cuando están en reposo y al momento de energizarlas se cierran impidiendo el paso.

4.1.3.4. Válvulas proporcionales

Las válvulas de acción directa también pueden ser válvulas proporcionales. En estas válvulas la cantidad de líquido que pasa a través de la válvula puede ser controlada cambiando la cantidad de corriente que atraviesa la bobina.

4.1.3.5. Bobinas solenoides

La bobina es el "motor" de la válvula; un flujo eléctrico crea un campo magnético permitiendo que la armadura fija atraiga el émbolo.

4.1.3.6. Aplicaciones

Algunas de las aplicaciones de electroválvulas son para:

- Agua
- Aire y gas inerte
- Vapor
- Aceites y gasolina
- Químicos
- Jarabes y agua desmineralizada
- Gas combustible

4.1.3.7. Control de energía de corriente alterna: control del ángulo de fase

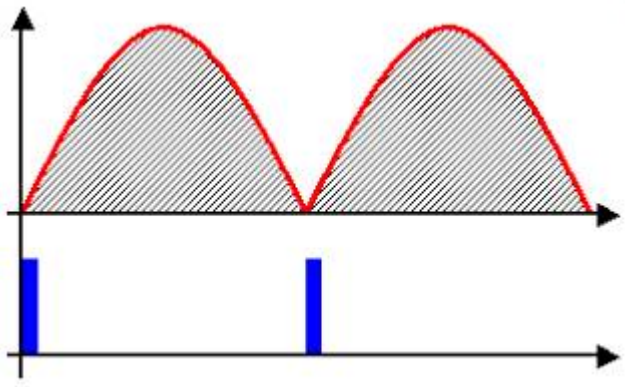


Figura 4.26. Control de ángulo de fase: tensión de salida controlada por la señal de control de puerta aplicada a un tiristor.

Control de ángulo de fase es un método de PWM aplicada a las tensiones de entrada de corriente alterna, por lo general el suministro de la red. Por supuesto, el suministro de AC podría ser de un transformador o cualquier otra fuente de corriente alterna, pero

el suministro de la red es la entrada más común - esto le da al método de control de ángulo de fase su mayor utilidad. La finalidad del control de ángulo de fase es controlar o limitar la potencia a la carga.

El dispositivo de la energía utilizada en los controladores de ángulo de fase es un tiristor - en su mayoría triacs o SCR. (Hay métodos de fase de control que emplea alta conmutación de frecuencia utilizando un MOSFET o IGBT, pero aquí voy a hablar de control de ángulo de fase con sólo tiristores). El flujo de energía a la carga es controlada por el retraso de la (tiempo de cocción cada semiciclo) ángulo de disparo al dispositivo de alimentación.

Se sabe que el tiristor es un dispositivo de bloqueo, que se activa mediante una señal de compuerta, la corriente es superior a la corriente de mantenimiento y el bloqueo actual, el tiristor permanece encendida, hasta que la corriente a través de ella se convierte en lo suficientemente baja (muy cerca a cero). El tiristor se apaga cuando la corriente a través de ella se convierte en cero, como sucede en la red de cruce AC cero. Esta es la conmutación línea natural. El supuesto aquí es que la carga es resistiva y tiene poca o ninguna inductancia. Por supuesto, esto no es siempre el caso, como cargas inductivas se utilizan a menudo. Sin embargo, se trabajara con esa hipótesis.

Por lo tanto, en el control de ángulo de fase, un impulso de puerta se envía al triac. Se envía a la vez entre un cruce por cero y el siguiente. Sin el impulso de puerta enviado al triac, justo después de cruce por cero, el triac está apagado y no fluye corriente a través de él. Después de un cierto tiempo, la señal de interrupción se da al triac y se enciende. El triac entonces permanece encendida hasta que la corriente a través de ella se convierte en cero (la conmutación línea natural). Esto es en el siguiente cruce cero. Por razones de simplicidad, y como generalmente debe ser, asumen que la corriente a través del triac (cuando en) es mayor que el bloqueo actual y la corriente de mantenimiento. Si no ya lo sabe, la corriente de enganche es la corriente que debe pasar por el triac justo después de que se enciende para asegurar que se trabe. La corriente de mantenimiento es el nivel de corriente a través del triac debajo del cual el triac se apagará. Por lo tanto, la suposición de que la corriente a través del triac es mayor que el enganche actual y la corriente de mantenimiento significa que el triac permanece en una vez que se dispara en. Permanece encendida hasta que la corriente a través de él es cero.

Esto significa que la tensión se suministra a la carga por una fracción del ciclo, determinado por el tiempo que el triac está encendida. El tiempo que el triac está encendido, está, a su vez, determina el tiempo de retardo entre el cruce por cero y la aplicación de la señal de compuerta del triac.

Así que, para resumir, ajustamos la tensión o la potencia entregada a la carga por el retraso de la señal de disparo al triac. Una cosa para recordar es que, la tensión suministrada y la potencia no se relacionan linealmente con el ángulo de fase de disparo.

Hay dos tensiones de aquí que nos ocupa - la tensión RMS y la tensión media. El voltaje RMS regula la salida de potencia a las cargas resistivas como bombillas incandescentes y calentadores resistivos. El valor medio se refiere a dispositivos que funcionan en el nivel medio de tensión. Esto es importante porque, cuando se prueba, el voltímetro registrará la tensión media - y no el voltaje RMS verdad - a menos que tenga un "voltímetro RMS real". La mayoría de los voltímetros de bajo costo no son verdaderos RMS metros pero responderán a los cambios de valor promedio.

Para aclarar por qué el poder y la tensión no están relacionados linealmente, vamos a examinar la fórmula que relaciona los dos:

$$P = VI = V * \frac{V}{R} = \frac{V^2}{R}$$

Así, suponiendo una resistencia constante, el poder es directamente proporcional al cuadrado de la tensión. Por lo tanto, si la mitad del voltaje, la potencia se redujo a la mitad no, pero se reduce a un cuarto de la potencia original. Una cuarta parte de alimentación con la tensión de media.

Fase de Diseño:

El paso por cero se realiza mediante el método de puente-optoacoplador como se había demostrado previamente.

Primero se verifica el cruce por cero. Después de que ocurre de cruce por cero, un pequeño retraso está presente antes de que se dispara el triac. Aquí, se utilizó 2 ms. Por lo tanto, el triac se dispara 2 ms después de producirse el cruce por cero. La señal de interrupción es 250µs eliminados después de eso. 250µs es tiempo suficiente para asegurar que el triac ha activado. A pesar de que se retira la señal de interrupción, el triac permanece activado hasta el siguiente cruce por cero, ya que es un dispositivo de enganche.

Ahora, ¿por qué quitar la señal de interrupción? Hemos detener en hasta el próximo cruce por cero. Bueno, eso sería demasiado trabajo. El problema no sería que, no habría altas pérdidas de conmutación del tiristor. La resistencia de control de puerta se disiparía inmensas cantidades de energía - todo sin razón, ya que el triac sería en incluso si se retira la señal.

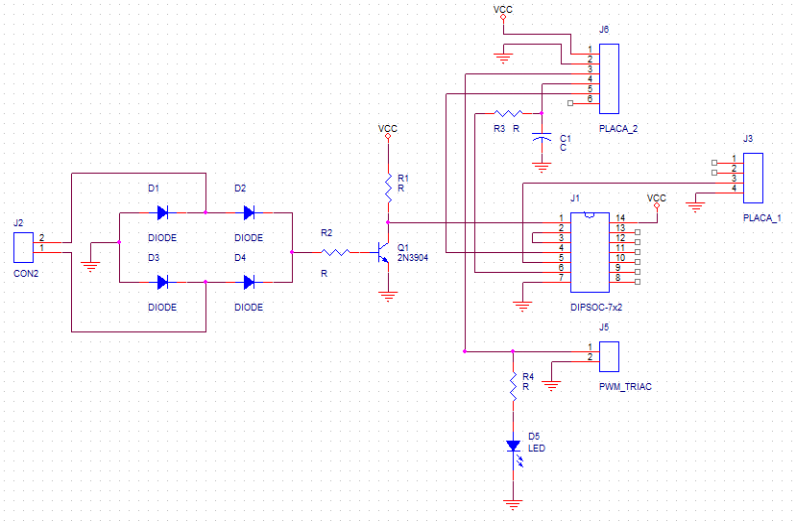


Figura 4.27. Diagrama de configuración del circuito

Se debe elegir R1 en función de las necesidades actuales de la puerta del triac. También debe tener una calificación suficientemente alta disipación de potencia. Por lo general, la potencia instantánea puede ser muy alta. Pero puesto que la corriente fluye a través de la resistencia por sólo 250US (1/40 de un ciclo medio de 50 Hz), la potencia media es lo suficientemente pequeño. Por lo general, 2W resistencias debería ser suficiente.

Supongamos que estamos usando un triac BT139-600. La corriente máxima requerida disparador es 35mA. Aunque la corriente de disparo típico es menor, debemos considerar la corriente máxima requerida gatillo. Esto es 35mA para cuadrantes I, II y III. Solo seremos disparando en los cuadrantes I y III. Así, es bien se considerara corriente 35mA.

Si no está seguro de qué cuadrantes son, he aquí una breve descripción:

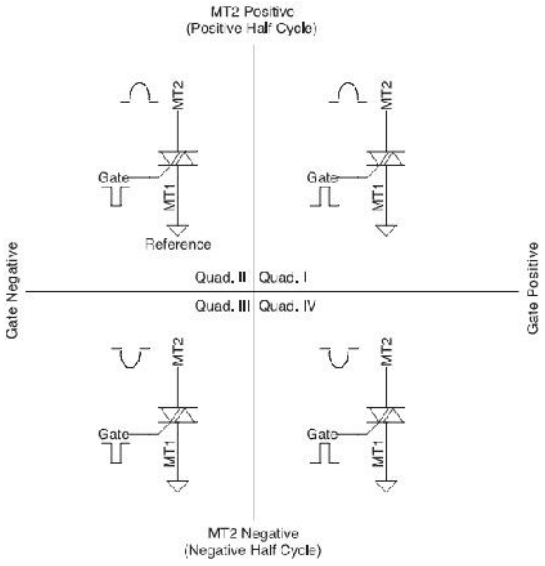


Figura 4.28. Cuadrantes de Activación Triacs

Si uno mira hacia atrás en la imagen 4.28, se verá que está conduciendo la puerta de MT2. Por lo tanto, podemos decir que, con respecto a MT1, MT2, cuando es positivo, por lo que es la puerta. Con respecto a MT1, MT2 cuando es negativo, por lo que es la puerta. Desde el diagrama anterior, se puede ver que estos dos casos están en los cuadrantes I y III.

El conductor en el circuito es el MOC3021. Esta es una fase aleatoria ópticamente controlador de salida aislado triac. Cuando el LED está encendido, el triac en el MOC3021 se enciende y se acciona el triac en el circuito principal. Es un conductor "fase aleatoria" lo que significa que puede ser accionado en en cualquier momento durante la señal de accionamiento, como se requiere para el control de ángulo de fase. Hay otros pilotos que sólo permiten unidad en el paso por cero.

Estos no se pueden utilizar para el control de ángulo de fase como el control de ángulo de fase requiere duro después de cruce por cero. Para garantizar que el triac está cerrada, el lado LED del MOC3021 debe ser conducido con un mínimo de corriente 15mA. La calificación máxima actual para el LED es 60mA. La calificación actual pico para el triac es 1A. Debes encontrar que nos hemos alojado dentro de estos límites en el diseño.

Aquí está la forma de onda de salida:

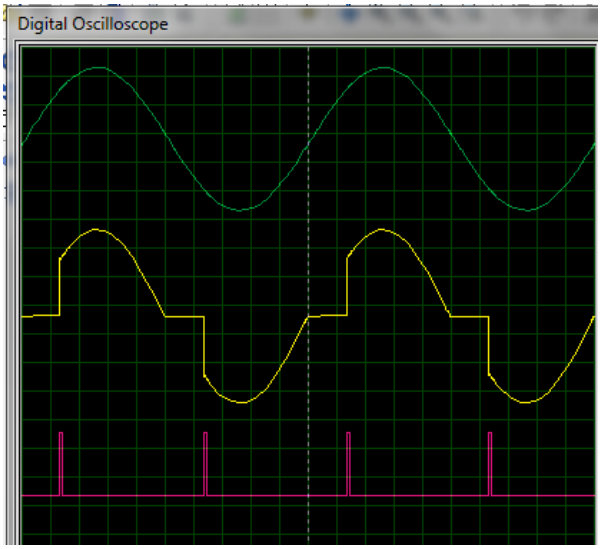


Figura 4.29. Triac disparando con retardo de 2ms (Verde: Entrada de CA, Amarillo: Salida de CA después del control de ángulo de fase, Rosado: señal control de puerta)

Se puede ver claramente que antes de aplicar la señal de excitación de compuerta, no hay salida (ilustrado por la línea amarilla plana) .Cuando se aplica la señal de activación de puerta, el triac se enciende. Hay una salida y el triac permanece en hasta el siguiente cruce por cero. Después de esto otra vez, no hay salida hasta la próxima señal de control de puerta se aplica.

Ahora se mostrara un par de formas de onda más, con otros retrasos iniciales.

Aquí, la puerta es accionado 1ms después de la de cruce por cero:

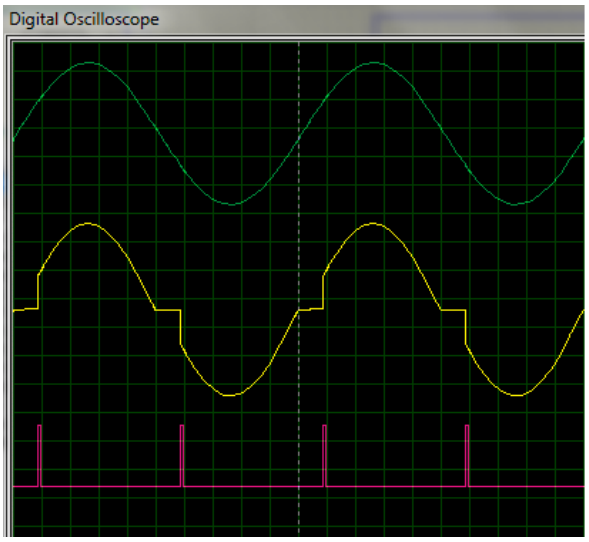


Figura 4.30. Triac disparando con retardo de 1ms (Verde: Entrada de CA, Amarillo: Salida de CA después del control de ángulo de fase, Rosado: señal control de puerta)

Aquí, la puerta es accionado 4 ms después del paso por cero:

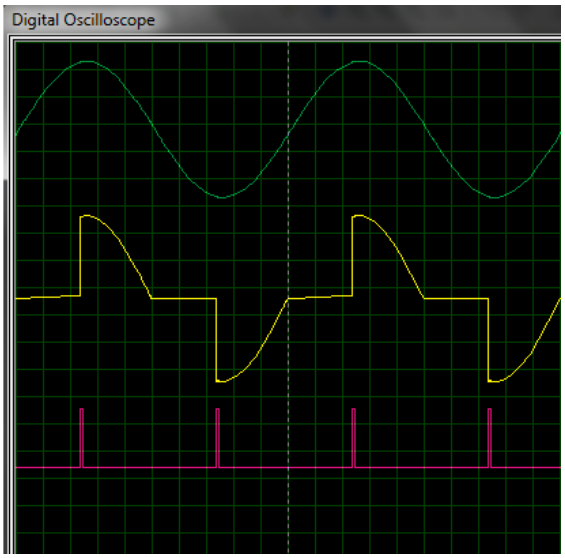


Figura 4.31. Triac disparando con retardo de 4ms (Verde: Entrada de CA, Amarillo: Salida de CA después del control de ángulo de fase, Rosado: señal control de puerta)

Aquí, la puerta es accionado 5ms después del paso por cero:

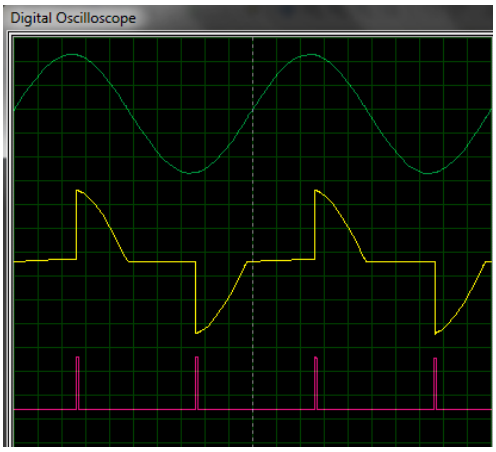


Figura 4.32. Triac disparando con retardo de 5ms (Verde: Entrada de CA, Amarillo: Salida de CA después del control de ángulo de fase, Rosado: señal control de puerta)

Aquí, la puerta es accionado 6 ms después del paso por cero:

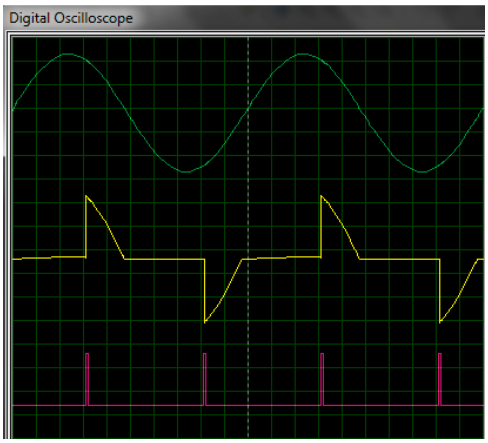


Figura 4.33. Triac disparando con retardo de 6ms (Verde: Entrada de CA, Amarillo: Salida de CA después del control de ángulo de fase, Rosado: señal control de puerta)

Ahora, se mostrara, cómo encontrar el valor eficaz de la tensión de salida.

En primer lugar, necesitamos saber cómo relacionar el retardo de disparo con ángulo de disparo. Sabemos que una onda sinusoidal completa es de 360 °. Es decir 2p radianes. Entonces tenemos que saber que el ángulo de disparo $\alpha = \omega t$, donde $\omega = 2\pi f$. Dado que, estamos trabajando con 50Hz aquí, f = 50 Hz. Por lo tanto, $\omega = 100\pi$. Sólo para poner a prueba esta relación, vamos a usar t = 0.020 segundos (20 ms). Por lo tanto $\alpha = 100\pi * 0.020 = 2\pi$, según lo dicho antes.

Por lo tanto, si estamos disparando a un retraso de 4 ms, es decir 4 ms después del paso por cero, el ángulo de disparo α = 100 π * (4/1000) = 0,4 π (en radianes obviamente).

El voltaje de salida RMS se encuentra desde la relación:

$$V_{rms} = V_s \left[\frac{1}{\pi} \left(\pi - \alpha + \frac{\sin(2\alpha)}{2} \right) \right]^{1/2}$$

Donde Vs = Voltage de alimentación RMS

Por lo tanto, si estamos disparando después de 4 ms, (α = 0,4 π), el voltaje RMS de salida es:

$$V_{rms} = 220 \left[\frac{1}{\pi} \left(\pi - 0.4\pi + \frac{\sin(0.8\pi)}{2} \right) \right]^{1/2} = 183.2V$$

Recordar, que al principio, se mencionó que la salida de tensión no se correlaciona linealmente con el ángulo de disparo. En este caso, la demora es de 4 ms. Por lo tanto, el triac está en el 60% del ciclo. Pero la tensión RMS de salida es 183.2V - 83% de la tensión de entrada. La falta de proporcionalidad directa es evidente aquí. La razón detrás de esto es la forma de la AC - sinusoidal.

4.2. DESCRIPCIÓN DEL CONTROLADOR PID EN MICROCONTROLADOR

Dado el amplio uso de los controladores PID en el ámbito industrial (control de potencia en motores de inducción, control de nivel, caudal y presión en procesos químicos entre otros), el uso de microcontroladores para el desarrollo de este tipo de aplicaciones ha tomado fuerza gracias a la incorporación de lenguajes de alto nivel que facilitan ampliamente este tipo de implementaciones, además de los bajos costos de adquisición de estos dispositivos.

4.2.1. CONTROLADOR PID

Es interesante señalar que más de la mitad de los controladores industriales que se usan hoy en día utilizan esquemas de control PID o PID modificado. Los controladores PID analógicos, son principalmente de tipo hidráulico, neumático, electrónico, eléctrico o sus combinaciones. En la actualidad, muchos de estos se transforman en formas digitales mediante el uso de microprocesadores. Se puede indicar que un controlador PID responde a la siguiente ecuación:

$$u(t) = K_p\, e(t) + \frac{K_p}{T_i} \int_0^1 e(t)\partial t + K_p\, T_d\, \frac{\partial e(t)}{\partial t}$$

donde e(t)es el error de la señal y u(t)es la entrada de control del proceso. Kp es la ganancia proporcional, Ti es la constante de tiempo integral y Td es la constante de tiempo derivativa.

En el dominio de la frecuencia, el controlador PID se puede escribir como:

$$U\,(S) = K_p\,(1 + \frac{1}{T_i\, S} + T_d\, S)E(S)$$

4.2.2. SINTONIZACION DE CONTROLADOR MEDIANTE ZIEGLER-NICHOLS

En lazo abierto, muchos procesos pueden definirse según la siguiente función de transferencia:

Donde los coeficientes K0, τ_o y γ_o se obtienen de la respuesta del sistema en lazo abierto a una entrada escalón. Se parte del sistema estabilizado en y(t) =y0 para u(t)=u0. Se aplica una entrada escalón de u0a u1 (el salto debe estar entre un 10% y un 20% del valor nominal) y se registra la respuesta de la salida hasta que se estabilice en el nuevo punto de operación.

Los parámetros se pueden obtener de la respuesta mostrada en la Figura 4.34:

$$\tau_0 = t_1 - t_0$$

$$\gamma_0 = t_2 - t_1$$

$$k_0 = \frac{y_1 - y_0}{u_1 - u_0}$$

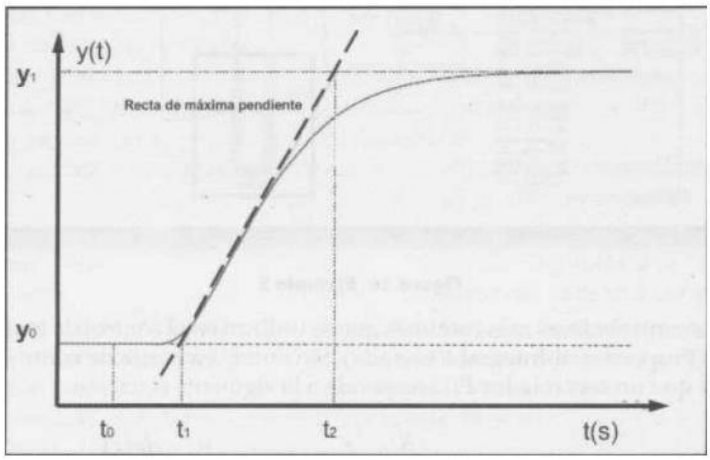


Figura 4.34. Respuesta de salida ante una entrada escalón

Según Ziegler-Nichols, la relación de estos coeficientes con los parámetros del controlador son:

$$K_p = 1.2\ \frac{\gamma_0}{k_0\ \tau_0} \qquad T_i = 2\tau_0 \qquad T_d = 0.5\tau_0$$

4.2.3. Controlador Digital PID

La función de Transferencia para el controlador PID digital se convierte en:

$$\mathrm{U(Z)} = K_p \left[1 + \frac{T}{T_i\ (1 - z^{-1})} + T_d\ \frac{(1 - z^{-1})}{T}\right]$$

La función de Transferencia discreta, también puede ser representada como:

$$\frac{U(z)}{E(z)} = a + \frac{b}{1 - z^{-1}} + c(1 - z^{-1})$$

Donde:

$$a = K_p \qquad b = \frac{K_p T}{T_i} \qquad c = \frac{K_p\ T_d}{T}$$

Existen distintas posibilidades de la realización práctica de un controlador PID, una de las más habituales es la realización en paralelo:

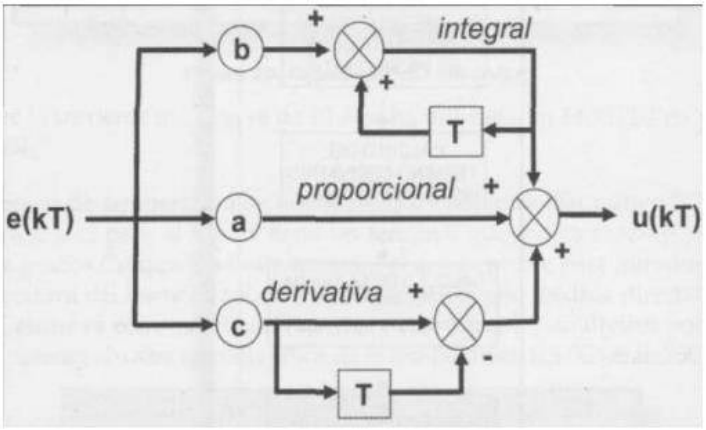


Figura 4.35. Diseño paralelo de controlador PID

CAPÍTULO 5

Software

5.1 SOFTWARE UTILIZADO EN EL DISEÑO DEL SISTEMA

Una vez presentado el hardware necesario para el Sistema de la Mini Planta de Control de Nivel a construir el siguiente paso será dar a conocer los programas que se utilizaron en este proyecto.

Dentro de los programas a mencionar se encuentran Keil uVision4 que es utilizado para editar y compilar el programa principal y distintas librerías en lenguaje C para el microprocesador
ARM.

5.2. Keil uVision4

El entorno de desarrollo uVision de Keil es una herramienta de carácter profesional de enorme calidad -que junto con algunas otras más- se ha convertido en un estándar para el desarrollo de aplicaciones basadas en las arquitecturas ARM-Cortex.

Figura 5.1. Keil uVision4

Las herramientas desarrolladas por Keil apoyan los microcontroladores más populares y son distribuidas en varios paquetes y configuraciones, dependiendo de la arquitectura.

- MDK-ARM: Kit de desarrollo de microcontroladores, para varios ARM7, ARM9 y dispositivos basados en Cortex-Mx.

- PK166: Kit de desarrollo profesional de keil, para dispositivos C166, XE166 y XC2000.
- DK251: Herramientas de desarrollo Keil 251, para dispositivos 251.
- PK51: Herramientas de desarrollo Keil 8051, para dispositivos Classic & Extended 8051.

Esta parte proporciona una introducción sobre software MDK-ARM (versión 4.1.0 y anteriores).

5.2.1. MDK-ARM

El MDK-ARM es una plataforma de desarrollo de software basado en ventanas que combina un robusto y moderno editor con un administrador de proyectos y lo hace una herramienta fácil. Integra todas las herramientas necesarias para desarrollar aplicaciones incluyendo el compilador C/C++, ensamblador de macros, enlazador/buscador, y un generador de archivos AXF. En la figura 4.2 muestra una vista general del software.

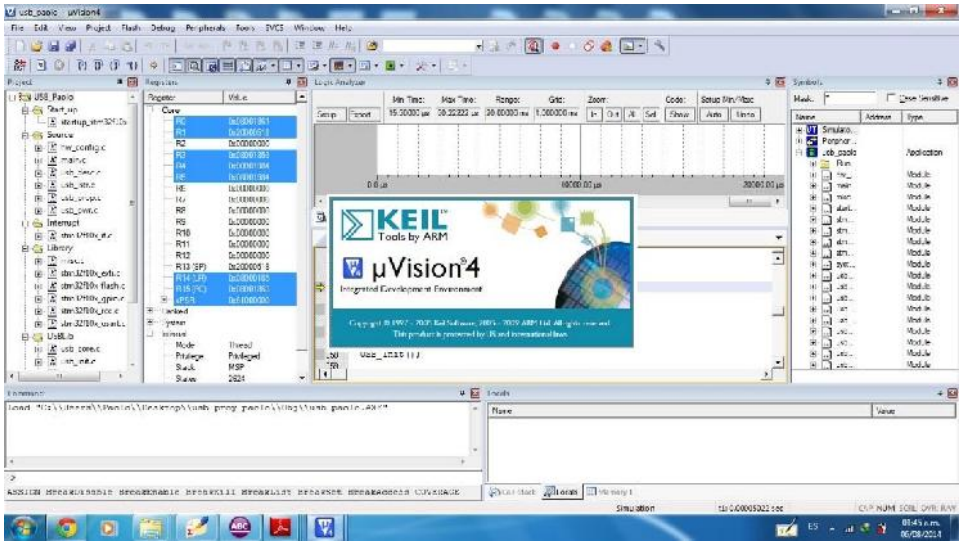


Figura 5.2: Ventana principal MDK-ARM

MDK-ARM ayuda acelerando el desarrollo de procesos de aplicaciones integradas proporcionando lo siguiente:

- Editor de código fuente con todas las funciones.
- Base de datos de los dispositivos para configurar la herramienta de desarrollo.
- Administrador de proyectos para la creación y mantenimiento de tus proyectos.

- Utilidad Make integrado para ensamblar, compilar y vincular sus aplicaciones embebidas.
- Dialogo para todas las opciones del entorno de desarrollo.
- Verdadero depurador a nivel de fuente y nivel ensamblador integrado con CPU de alta velocidad y un simulador de periféricos.
- Utilidad de programación Flash para la descargar del programa de aplicación en Flash ROM.
- Enlace a manuales, ayuda en línea, hoja de datos de dispositivos, y guía de usuarios.

5.2.2. Conceptos de las ventanas de diseño

Nosotros podemos configurar el entorno de trabajo de uVision según nos sea conveniente.

Sin embargo se definirá tres áreas más importantes. Estas definiciones ayudarán para el entendimiento futuro de comentarios, ilustraciones e instrucciones.

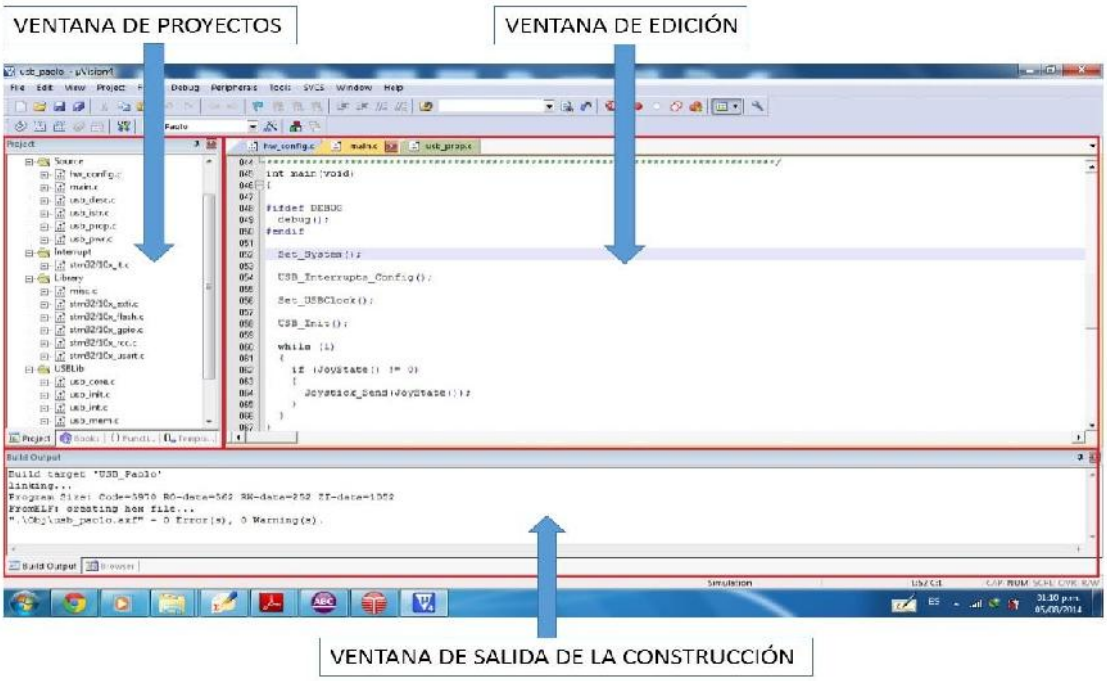


Figura 5.3: Áreas de trabajo

El área de la **ventana de proyectos** es la parte de la pantalla en la cual, por defecto, el proyecto, las funciones, los libros y registros son mostrados.

Con el área de la **ventana de edición**, nosotros somos capaz de cambiar el código fuente, ver la performance y análisis de información, y verificar el código de desmontaje.

El área de la **ventana de salida** proporciona información relacionada con la depuración, memoria, símbolos, llamada de pila, variables locales, comandos, buscador de información y encuentra los resultados en archivos.

Si, por alguna razón, tú no puedes ver una ventana en particular y has tratado de mostrar/ocultar esto varias veces, por favor invoque el diseño por defecto de uVision través del menú Windows - Reset Current Layout.

5.2.3. Redistribución de la zona de trabajo

La zona de trabajo puede cambiarse de aspecto y de organización. Para ello es necesario actuar sobre alguna o algunas de las restantes ventanas, lo que origina un cambio conjunto de las distribuciones de todas las ventanas y zona de trabajo. Los procesos que se van a comentar se pueden aplicar sucesivamente con diferentes ventanas o pestañas una tras otra.

La manera de proceder es la siguiente:

- Primero, pinche en la barra superior de la ventana que se desee reorganizar (o en la pestaña oportuna en el caso de una organización en pestañas y que solo se desee reubicar esa pestaña en concreto y no todas las de la ventana). En el ejemplo, la pestaña **Templates**.

- Manteniendo pulsado el botón izquierdo del ratón, arrástresela. En ese preciso momento aparecerán unos símbolos u operadores de organización, tal y como se aprecia en la figura 5.4.

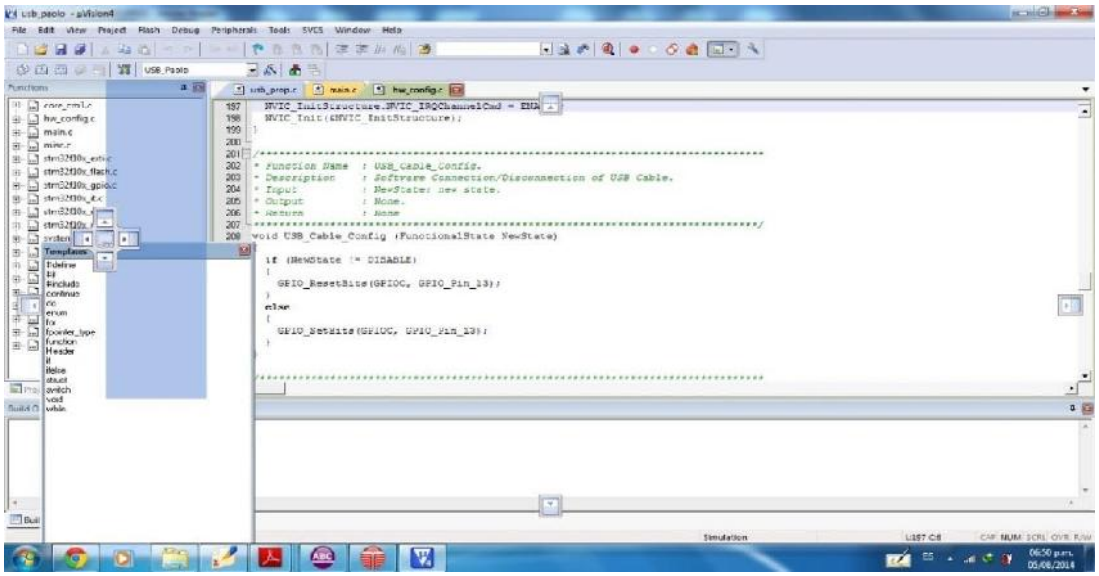


Figura 5.4: Reorganización de ventanas

5.2.4. Modos de uVision

uVision opera en dos modos: **Modo de Construcción y Modo de Depuración**. Ajustes de pantalla, ajustes de barra de herramientas y opciones de proyecto son almacenados en el contexto del modo.

La barra de herramientas de **Archivo** está activa en todos los modos, mientras que la barra de herramientas de depuración y construcción está mostrados solamente en sus respectivos modos. Botones, íconos y menús son activados si son relevantes para un modo específico.

- **Modo de Construcción:** Es el modo de trabajo estándar. En este modo escribimos nuestra aplicación, configuramos el proyecto, establecemos preferencias, seleccionamos el hardware de destino y el dispositivo; compilaremos, enlazaremos, y ensamblaremos el programa, corregiremos errores, y estableceremos ajustes generales válidos para toda la aplicación.

- **Modo de Depuración:** En este modo nosotros también podemos cambiar algunas opciones generales y editar archivos de código fuente, pero esos cambios solo se harán efectivos después de que haya cambiado de nuevo a modo de construcción, y reconstruir la aplicación. Los cambios de ajustes de depuración son inmediatamente efectivos

5.2.5. Barra de menú y de herramientas

La figura 5.5 muestra en la parte superior la barra de menú, mediante el cual se puede acceder a todas las opciones de uVision, y debajo de ella se encuentra la barra de herramientas, con los íconos de las funciones más usuales. Estos íconos suponen un atajo alternativo para realizar tales tareas.

Figura 5.5: Barra de menú y de herramientas

Es importante advertir que si ya se hubiese trabajado con algún proyecto y se hubiese modificado la apariencia y forma de las ventanas de la interfaz, entonces al entrar en uVision4 se mostrará la interfaz exactamente igual a como quedó al cerrarse la última vez.

5.2.6. Ventana de proyecto

La ventana de proyecto muestra información acerca del proyecto actual. Las fichas en la parte inferior de esta área proporcionan acceso a:

- **Proyect:** Estructura y gestiona el proyecto. Agrupa los archivos para mejorar la visión general del proyecto.

- **Functions:** Muestra las funciones del proyecto. Encuentra y navega rápidamente entre funciones del código fuente.

- **Registers:** Registros del microcontrolador. Solamente son disponible durante la depuración.

- **Templates:** Plantillas de bloques de texto de uso frecuente. Doble clic en la definición para insertar el texto predeterminado en la posición del cursor.

- **Books:** Libros específicos para el IDE uVision, el proyecto y a veces del microcontrolador usado. Configura y añade sus propios libros a cualquier sección.

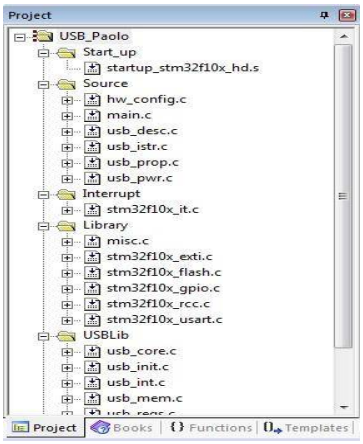


Figura 5.6: Ventana de proyecto

5.2.7. Ventana de Edición

La ventana de edición es usada para:

- Escribir, editar, y depurar archivos fuente.
- Establece puntos de ruptura (breakpoint) y marcas de libro (bookmarks).

- Establece opciones de proyecto e inicializa el sistema de destino mediante potentes asistentes de configuración.
- Ver el código de desmontaje y traza durante la depuración.

Típicamente, esta área contiene el editor de texto con los archivos de código fuente, la ventana de desmontaje, el analizador de rendimiento, y el analizador lógico.

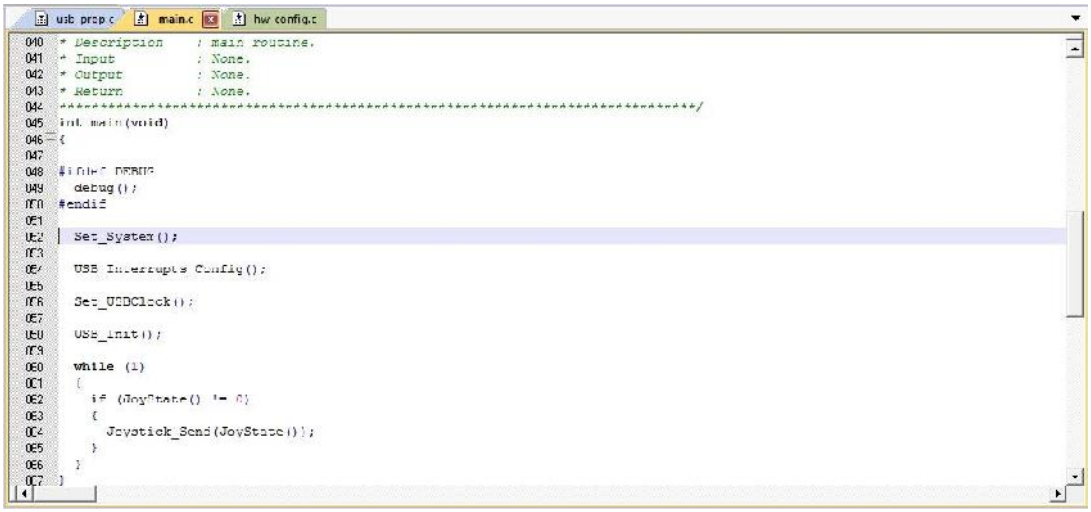


Figura 5.7: Ventana de edición

5.2.8. Editor de configuración

Configura las opciones del editor, colores y fuentes, palabras clave definidas por el usuario, y plantillas a través del diálogo de configuración.

Podemos invocar este diálogo vía el **Menú Edit - Configuration**.

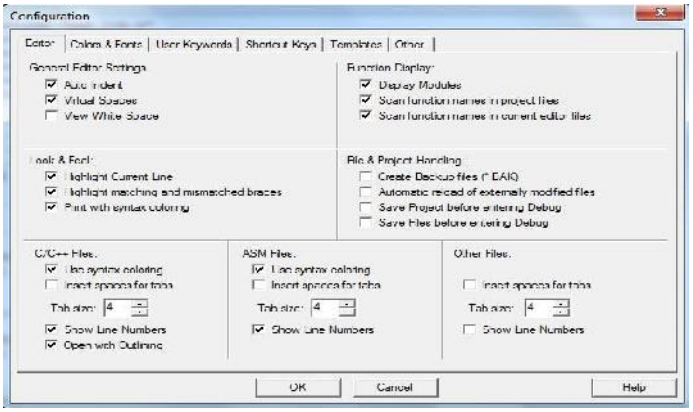


Figura 5.8: Diálogo de configuración

5.2.9. Ventanas de Salida

Por defecto, las ventanas de salida son mostradas en la parte inferior de la pantalla de uVision e incluye:

- La **Ventana de Salida de Construcción** incluye errores y avisos del compilador, ensamblador, y enlazador.
- La **Ventana de Comandos** permite ingresar comandos y evaluar.
- La **Ventana de Búsqueda de Archivos** permite hacer doble clic en un resultado para localizar el código fuente que desencadeno el mensaje.
- La **Ventana Serial y Uart** muestra I/O de información de los periféricos.
- La **Ventana de Llamada de Pila** permite seguir el árbol de llamadas de programa.
- La **Ventana Local** muestra información acerca de variables locales de la función actual.
- La **Ventana de Reloj** suministra una conveniente manera de personalizar un conjunto de variables que les gustaría investigar. Objetos, estructuras, uniones, y arreglos pueden ser monitoreados en detalle.
- La **Ventana de Símbolos** es una opción útil para localizar definiciones de objetos.
- La **Ventana de Memoria** permite revisar valores en el área de memoria. Define las direcciones preferidas para ver los datos.
- La **Ventana de Buscador de Fuente** ofrece un modo rápido para encontrar ocurrencias y definiciones de objetos. Use sus criterios de búsqueda para reducir la salida.

5.2.10. Ayuda en Línea

uVision incluye muchas páginas de manuales en línea y ayuda sensible al contexto. La principal ayuda del sistema es disponible desde el Menu Help.

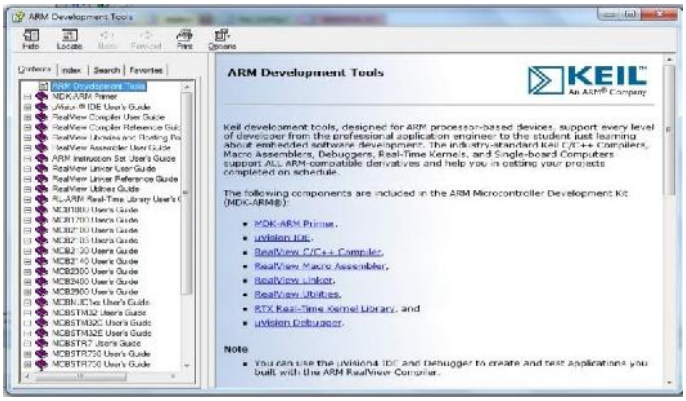


Figura 5.9: Ayuda en línea

5.3. Flash Loader

El Flash Loader nos sirve para escribir programas en nuestro microprocesador generalmente con extensión .hex y para leer los programas que hayan sido escritos con anticipación.

5.3.1. Requisitos del Sistema

Para utilizar el Flash Loader con el sistema operativo Windows, debe estar instalado en el PC una versión reciente de Windows, como Windows 98, Millenium, 2000, XP, Vista o Windows 7. La versión del sistema operativo Windows instalado en el equipo puede ser determinado pulsando el botón derecho en el icono "Mi PC" en el escritorio, a continuación, hacer clic en la opción "Propiedades" en el menú emergente que aparece. El tipo de sistema operativo aparece en el cuadro de diálogo "Propiedades del sistema" bajo la etiqueta de "sistema" como se muestra en la figura 5.10.

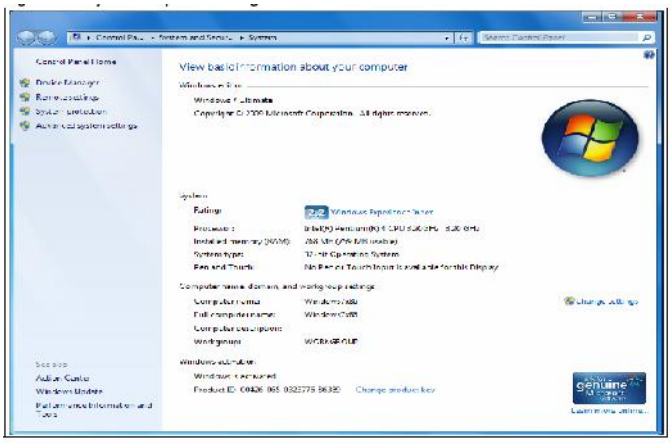


Figura 5.10: Cuadro de diálogo de Propiedades del sistema

Para fines de comunicación, es necesario comprobar que tiene un puerto COM disponible (RS232) si la aplicación implementa la interfaz UART.

Para comprobar que se dispone de una interfaz disponible (COM), haga clic en el icono "Mi PC" en el escritorio y seleccione "Propiedades" en el menú emergente. Aparecerá el cuadro de diálogo "Propiedades del sistema". Haga clic en la pestaña "Hardware", y luego en el botón "Administrador de dispositivos" para mostrar la configuración de hardware del sistema. Puertos COM disponibles se agrupan bajo los "Puertos (COM & LPT)" nodo en el árbol de hardware como se muestra en la Figura 5.11.

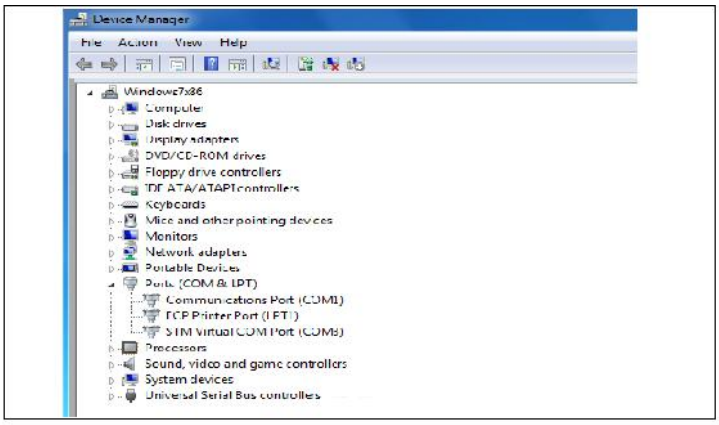


Figura 5.11: Ventana Administrador de dispositivos

Es interesante conocer las capacidades del puerto COM. Para averiguarlo, haga clic derecho sobre el elemento Puerto de comunicación (COMx) y luego haga clic en "Propiedades" para abrir la ventana Propiedades. Seleccione la pestaña "Configuración de puerto", a continuación, haga clic en la flecha situada junto a los "bits por segundo" cuadro combinado para conocer las velocidades de transmisión admitidas por el puerto.

5.3.2. Instalación de Software

Si una versión anterior está instalado en su computadora, retírelo utilizando el "Agregar o quitar programas" del servicio en el "Panel de control".
Ejecute el archivo Setup.exe proporcionado: el asistente de instalación le guiará a través de la instalación de la aplicación de demostración cargador de Flash en el equipo como se muestra en la Figura 5.12 y la Figura 5.13 (Debe aceptar el contrato de licencia para instalar el software).

Figura 5.12: Asistente de Instalación

Figura 5.13: Acuerdo de licencia

Una vez que el software se ha instalado correctamente, haga clic en el botón "Finalizar". El archivo version.txt que contiene las nuevas notas de la versión se abrirá automáticamente en la aplicación Bloc de notas nativa Microsoft®-. Bloc de notas de clausura pondrá en marcha el cargador manifestante Flash, si las casillas de verificación se guardan por defecto en el asistente de instalación.

5.3.3. Instalación de hardware

Como el cargador demostrador Flash es capaz de comunicarse a través de la interfaz UART, el dispositivo debe estar conectado a un puerto COM PC de repuesto en el caso de una comunicación UART.

5.3.4. Descripción de la interfaz de usuario

El cargador manifestante flash está diseñado como una aplicación de asistente. Se estructura en seis pasos, el:

- Configuración de la conexión
- Estado de Flash

- Información del dispositivo
- Operación elección
- Edición de bytes Opción
- Progreso de la operación

a) Paso 1

Ejecutar la aplicación Flash Loader en el menú "Programas" (conexión con el dispositivo no se ha hecho aún) a continuación, asegúrese de que el dispositivo está conectado a su PC y reiniciarlo para reiniciar el código del gestor de arranque de la memoria del sistema.

Esta etapa consiste en la selección de la interfaz de conexión UART y sus ajustes relacionados. Establezca los parámetros de conexión (nombre de puerto, velocidad de transmisión y el tiempo de espera, etc.), como se muestra en la figura 5. Para una configuración óptima para la interfaz UART, ajuste "Baud Rate" al 115200 bits por segundo y "Tiempo de espera (s)" para 5 segundos.

Asegúrese de que los pasadores de configuración de arranque se establecen correctamente, a continuación, haga clic en "Siguiente" para continuar. Si una conexión se ha establecido, el asistente se mueve al siguiente paso, de lo contrario se muestra un cuadro de mensaje que indica el error que se produjo.
Errores Posibles mensajes:

- "No se puede abrir el puerto COM": se muestra este mensaje si no se encuentra el puerto COM seleccionado o si ya está siendo utilizado por otro proceso.
- "dispositivo no reconocido": este mensaje se muestra si el valor recibido es diferente de 0x79. Restablecimiento del dispositivo puede resolver el problema.
- "No hay respuesta del objetivo": este mensaje aparece cuando no hay respuesta desde el objetivo. Indica que el gestor de arranque de la memoria del sistema no es funcional. Verifique la configuración de arranque y compruebe que el microcontrolador utilizado contiene el código del gestor de arranque.

Nota: El argumento de tiempo de espera es el período de tiempo después de que una petición de lectura desde el puerto serie se cancela si no se reciben datos. El valor recomendado es de 5 segundos, pero depende del entorno utilizado, al igual que el rendimiento del hardware.

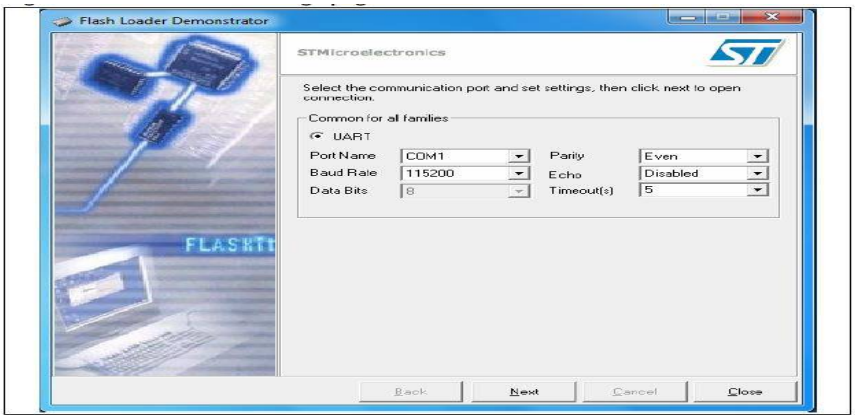


Figura 5.14: Configuración de la conexión

Nota: "Echo" está presente en la versión 2.1.0 del demostrador cargador de Flash para el apoyo de algunos de los dispositivos que utilizan STM8 LIN echo hacia atrás la emulación a través del protocolo UART. Si no se utilizan estos dispositivos, esta opción debe mantenerse inhabilitado.

b) Paso 2

En el segundo paso, la conexión se ha establecido y ha comenzado la comunicación. Consiste en mostrar el estado de la memoria flash. Este estado puede ser protegido leer, en cuyo caso se desactiva el botón "Siguiente" hasta que la protección de lectura se elimina haciendo clic en el botón "Quitar la protección".

Nota: Al hacer clic en el botón "Quitar la protección" no sólo leerá-desproteger la memoria flash, también borrará todas sus páginas.

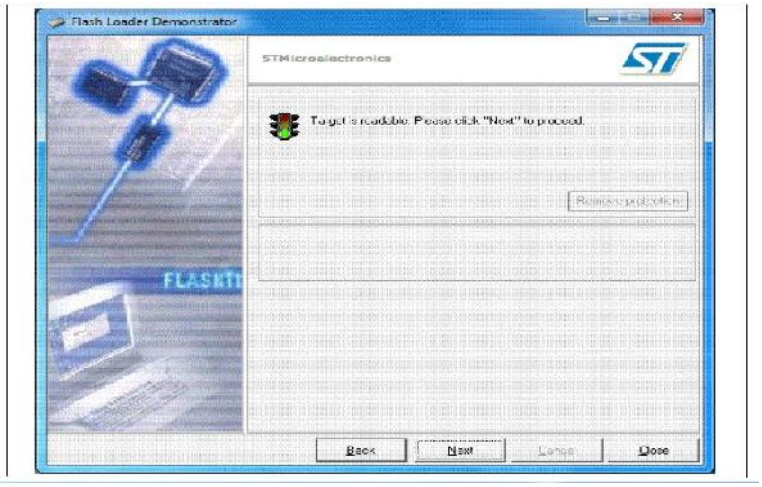


Figura 5.15: Estado de Flash

c) Paso 3

En este paso, el Asistente muestra la información del dispositivo disponibles, tales como el ID de destino, la versión de firmware, el dispositivo compatible, el mapa de la memoria y el estado de protección de memoria.

Seleccione el nombre de destino en el cuadro de destino, como se muestra en la Figura 5.16 y la Figura 5.17, a continuación, haga clic en "Siguiente" para continuar.

Figura 5.16: Información de dispositivos - ejemplo STM32

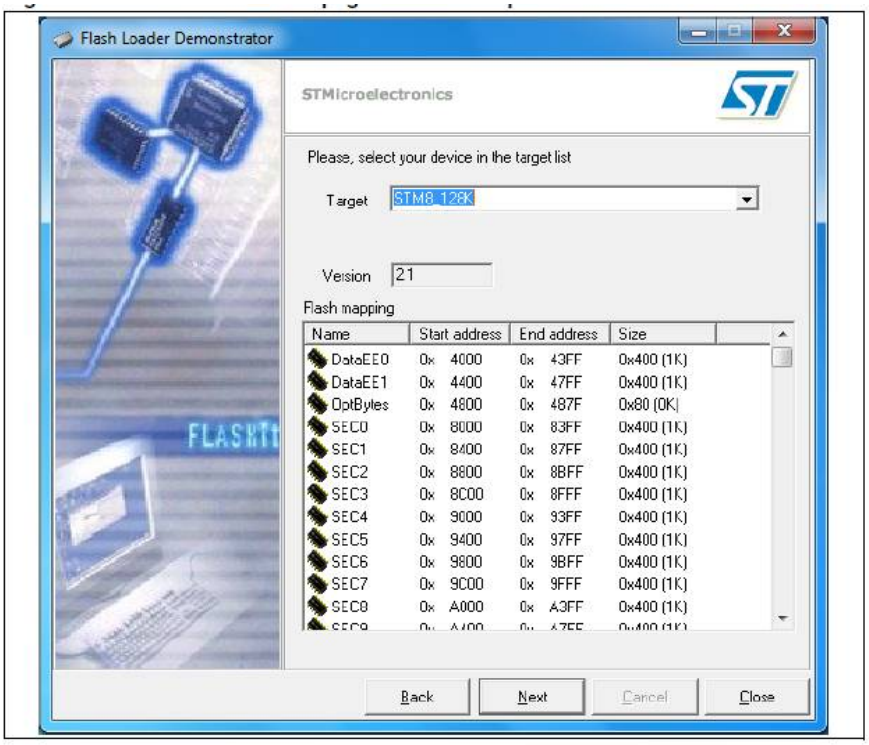


Figura 5.17: Información de dispositivos - ejemplo STM8

d) Paso 4

En este paso, seleccione la -Eliminación operación solicitada, descargar, cargar o Desactivar / Activar la protección Flash o bytes opción Editar - y establecer los parámetros relacionado

1. **Borrar**
   a) Seleccione la opción "Todo" para borrar toda la memoria

   b) Seleccione la opción "Selección" para personalizar la operación de borrado. Haga clic en el botón "..." para abrir la ventana de diálogo de asignación de memoria. A continuación, compruebe las páginas que desea borrar y haga clic en "Aceptar

2. Descargar

- Haga clic en el botón Examinar relacionados para abrir un binario, hexadecimal o archivo S19 Motorola.
- Si el archivo cargado es un archivo binario, la dirección de descarga es la dirección de salida de la primera página y el campo "@" se sigue editable para aceptar los cambios.

- Si el archivo cargado es de un hexadecimal o un archivo S19 Motorola, la dirección de descarga es la dirección de inicio del primer registro en el archivo, y el campo de "@" es de sólo lectura.

- Marque la casilla de verificación "Verificar" para iniciar el proceso de verificación.

- Se terminó Descargar operación.
- Marque "Saltar al programa de usuario" para iniciar el programa descargado.
- Marque "Optimizar" para filtrar los paquetes FFs (256 bytes).
- Marque "Aplicar bytes de opciones", a continuación, busque el archivo byte opción creado por la operación "Editar bytes de opciones". Los valores en el archivo seleccionado se aplicarán al dispositivo después de la descarga.

3. Subir

- Haga clic en el botón Examinar relacionados para seleccionar qué binario, hexadecimal o archivo S19 Motorola almacenarán los datos cargados.

4. Desactivar / Activar la protección de Flash

- Seleccionar las opciones de los dos menús desplegables para compensar el comando deseado (Habilitar Leer protección, Desactivar Leer protección, Activar la protección de escritura, Desactivar Escriba protección). Todos los comandos de protección se aplicarán a todas las páginas de memoria Flash a excepción de la protección contra escritura Activa, que se puede personalizar. Esto se hace haciendo clic en el botón "..." para seleccionar las páginas que ser protegido contra escritura.

5. Edite bytes de opciones

- Si es necesario establecer los bytes de opción, marque la opción a continuación, haga clic en "Siguiente" para pasar a la página de edición de bytes opción (Paso 5 Figura 5.20).

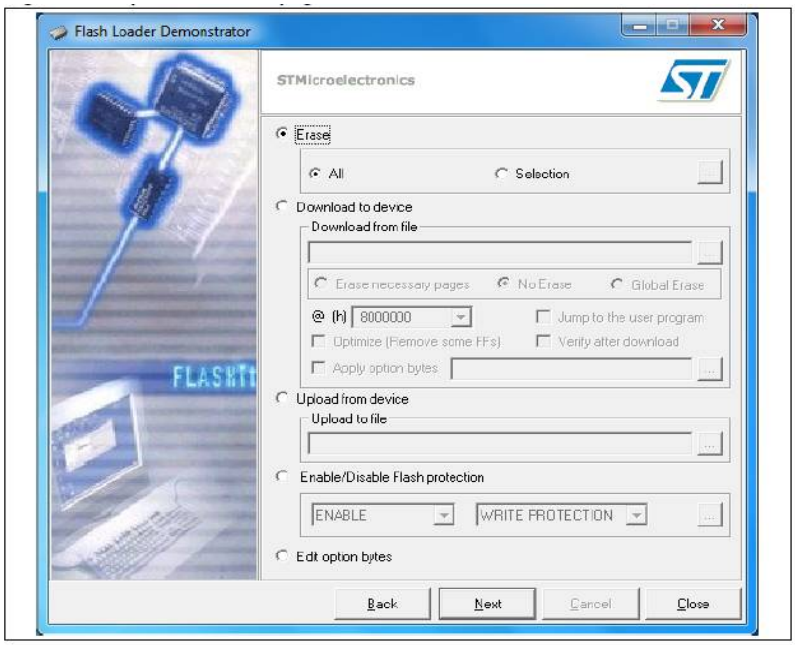


Figura 5.18: Elección de funcionamiento para STM32

Figura 5.19: Elección de funcionamiento para STM8

e) Paso 5

Nota: Este paso sólo se aplica a los dispositivos STM32. No hay paso 5 para dispositivos STM8.
La última página del Asistente depende de la operación seleccionada en el paso4.

a) Caso de una "opción Editar bytes" operación:
Se muestra la página edición de bytes Opción. Contiene los valores de bytes de opciones actuales cargados desde el dispositivo: RDP, USUARIO, Data0, Datos1, WRP0, WRP1, WRP2 y WRP3.
Para más detalles, por favor refiérase a la sección de cargador de bytes Opción en el "manual de programación STM32F10xxx flash" (PM0042 disponible de www.st.com).
Este paso da la posibilidad de aplicar los valores de bytes de opciones editados, cargarlos desde el dispositivo y guardarlos en un archivo.

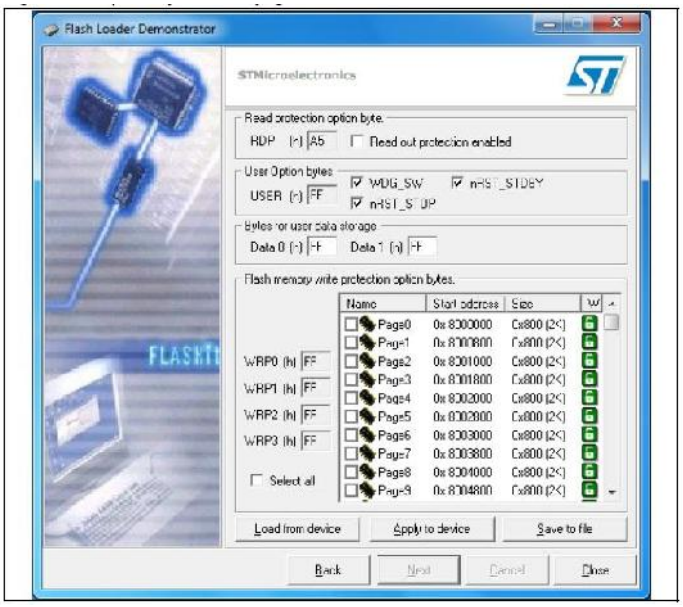


Figura 5.20: Edición de Bytes

b) Caso de cualquier otra operación:
Se muestra la página de funcionamiento. Se da el tamaño de los datos para ser descargado o cargado, el porcentaje completado y la duración de la operación como se ilustra en la Figura 5.21.

- Si la operación tiene éxito, la barra de progreso es de color verde. Si se produce un error, la barra se vuelve de color rojo y se muestra el error.

- Para detener la operación, haga clic en el botón "Cancelar".

- Si el "Salto al programa de usuario" casilla de verificación se comprobó en el paso anterior (paso 4), y el programa de usuario se ha descargado con éxito, se pierde la comunicación con el gestor de arranque de la memoria del sistema.

En consecuencia, el botón "Atrás" se redirige a la página "Ajustes de conexión" (paso 1) para evitar el lanzamiento de una nueva operación.
Si el "Salto al programa de usuario" casilla de verificación no se comprobó en el paso 4, el botón "Atrás" sigue activo y se puede volver al paso 4 y seleccione una nueva operación.

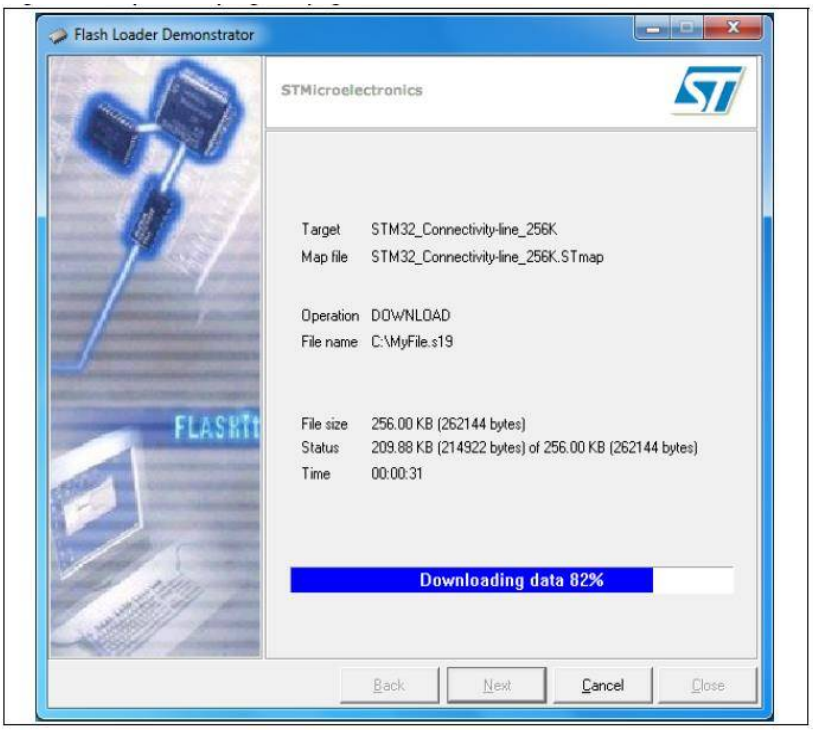


Figura 5.21: Progreso de la Operación

CAPÍTULO 6

Desarrollo del Proyecto

6.1. INTRODUCCIÓN

En este capítulo se describe los requerimientos previos, diseños y construcción para el Sistema de la Mini Planta de Control de Nivel incluyendo el acondicionamiento de las E/S, el acoplamiento de módulos, control y la comunicación.

La descripción del programa principal y archivos de interface se describirán con diagramas de flujo para su mejor interpretación.

6.2. DESCRIPCIÓN DEL PROCESO

Esta planta de control de nivel, como su propio nombre lo indica controla el nivel de agua, haciendo uso del control digital realimentado.

El CPU de este proceso es PLACA N° 01 Stm32-V3 representado por LC en el diagrama de flujo, que procesa las señales obtenidas del sensor Modulo SR-04T (LT en el diagrama).

El funcionamiento de este sensor es simple: emite un pulso de ultrasonido que rebota sobre el agua y calcula la distancia midiendo el tiempo que transcurre entre la emisión del sonido y la percepción del eco.

La PLACA N° 01 compara la variable del proceso (PV) que vendría ser la altura del nivel de agua con el valor deseado (SP) y envía su respuesta al variador de velocidad STM32-V3 y a la válvula BPX2-97A. El variador de velocidad a su vez controla a la electroválvula, aumentando y/o disminuyendo el caudal de entrada al tanque.

El desplazamiento del obturador de la válvula de control es controlado directamente por el CPU de este proceso, a fin que la diferencia entre el SP y PV sea la mínima posible.

En este sistema también podemos agregar perturbaciones externas a través de las válvulas manuales. Pero la característica principal de este tipo de control es que le hace inmune a las perturbaciones, obteniendo la altura deseada que es establecida en la Placa N° 01.

Figura 6.1. Imagen Lateral de la Mini Planta de Nivel

6.2.1. Diagrama de flujo del proceso

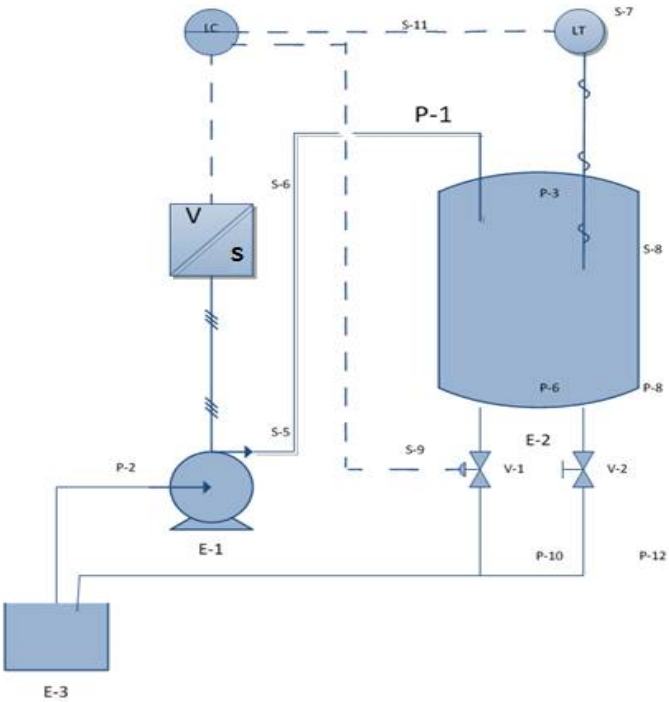


Figura 6.2. Diagrama flujo del proceso general de la planta de nivel, elaborado en Microsoft Visio 2010

DENOMINACIÓN	DESCRIPCIÓN	MARCA/MODELO
E-1	Electroválvula	
E-2	Tanque cerrado	
E-3	Tanque abierto	
V-1	Electroválvula	BPX2-97A
V-2	Válvula manual	
V/S	Variador de Velocidad	STM32-V3
LT	Sensor ultrasónico	Modulo SR-04T
LC	Placa N° 01	STM32-V3
-------	Señal eléctrica	
—//—	Señal neumática	
~o~ ~o~	Señal electromagnética	
———	Señal mecánica	

6.2.2. Diagrama de bloque del sistema de control de nivel

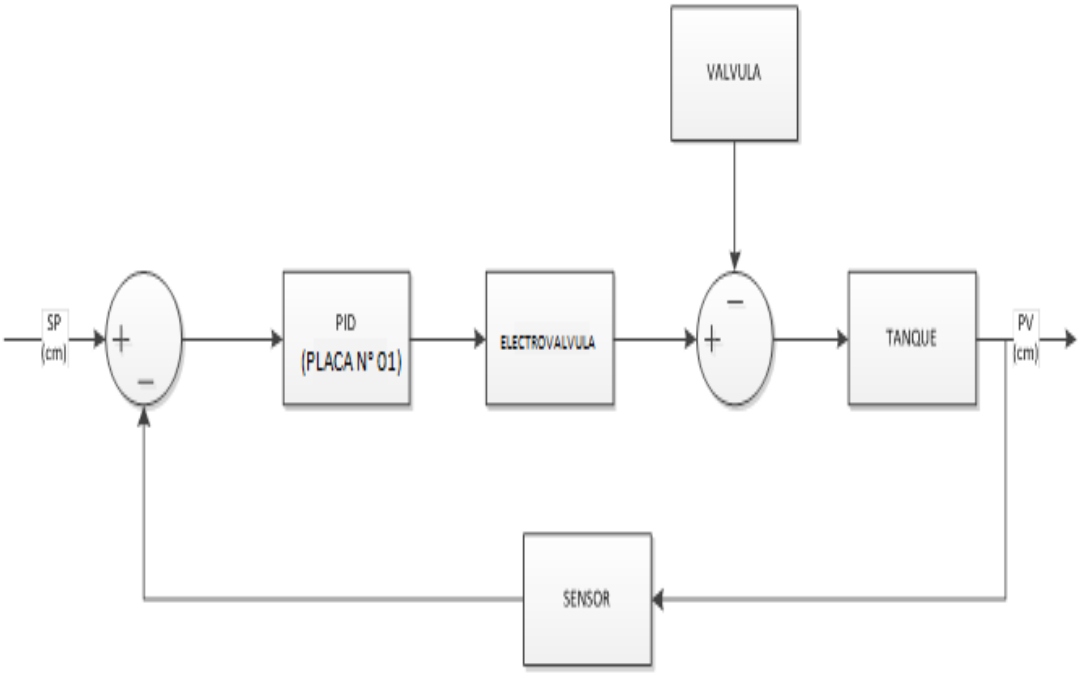


Figura 6.3. Diagrama del bloques del proceso general de la planta de nivel, elaborado en Microsoft Visio 2010

6.3. Modelo Matemático de los procesos de la Mini Planta de Nivel

6.3.1. Modelo matemático del control PID:

Uno de los controladores más utilizados en control de procesos industriales es el denominado controlador de tres términos o controlador PID. Este controlador tiene una función de transferencia

$$G_c(s) = K_p + \frac{K_i}{s} + K_D s.$$

El controlador proporciona un término proporcional, un término integral y un término derivativo. La ecuación para la salida en el dominio del tiempo es:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_D \frac{de(t)}{dt}.$$

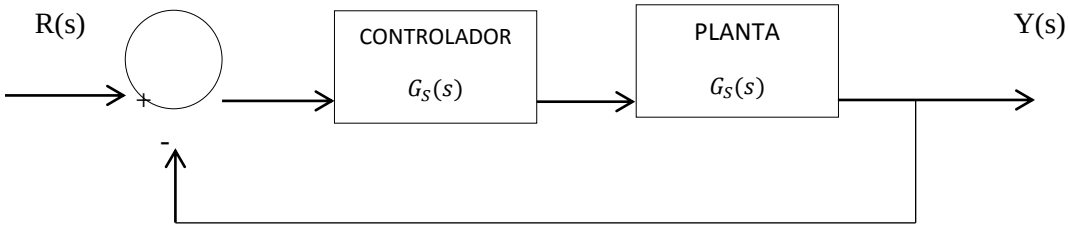


Figura 6.4. Sistema en lazo cerrado de un controlador

6.3.2. Modelo matemático del motor ac:

Los motores de corriente alterna se describen por dos ecuaciones diferenciales: El par (T (t)) es descrito por

$$T(t) = K.v(t) - m\frac{d}{dt}\theta(t)$$

Donde K es una constante, v (t) es el voltaje proveniente del motor, θ(t) es la posición angular del motor, y m es descrito por:

$$m = \frac{\text{"Puesto de torque (un valor nominal de tensión)"}}{\text{"velocidad sin carga (a tensión nominal)"}}$$

Dónde: Puesto de torque (a tensión nominal) y velocidad sin carga (en tensión nominal) son las características específicas de cualquier motor de AC. El torque también es descrito por:

$$T(t) = I\frac{d^2}{dt^2}\theta(t) + B\frac{d}{dt}\theta(t)$$

Igualando las dos ecuaciones del motor de AC, suponiendo condiciones iniciales nulas, y luego tomar la transformada de Laplace de la ecuación resultante, la función de transferencia de un motor de CA se encuentra para ser:

$$\frac{\Theta(s)}{V(s)} = \frac{K_m}{s(\tau s + 1)}$$

Dónde: $K_m = \frac{K}{m+B}$

Es una constante, y:

$$\tau = \frac{I}{m + B}$$

Es el tiempo constante del motor. Este es el análisis de la función de transferencia para el motor AC.

6.3.3. Modelo matemático de la válvula:

En cualquier estudio de válvulas de control y sus características, se deben tomar en cuenta dos partes de la válvula en forma especial: Primero, el cuerpo de la misma, sus aspectos geométricos y los materiales de construcción y en segundo lugar, el macho o tapón de la válvula, su geometría y sus materiales de construcción. La geometría combinada del cuerpo y el tapón determinan las propiedades de flujo de la válvula.

La mayoría de las válvulas operan por medio de un actuador de posición lineal o alguna modificación de este tipo de actuador. Estos actuadores colocan el macho de la válvula en el orificio, en respuesta a una señal proveniente del controlador automático o a través de un ajuste mecánico manual.

Una válvula neumática siempre tiene algún retraso dinámico, el cual hace que el movimiento del vapor no responda instantáneamente a la presión aplicada desde el controlador. Se ha encontrado que la relación entre el flujo y la presión para una válvula lineal puede a menudo representarse por una función de transferencia de primer orden; esto es:

$$G_V(s) = \frac{Q(s)}{U(s)} = \frac{K_v}{\tau_V s + 1}$$

Donde *Q(s)* = variable manipulada

- *U(s)* = señal proveniente del controlador (presión o mA) y actúa sobre la válvula
- K_V = constante de válvula (ganancia al estado estacionario)
- τ_V = Constante de tiempo de la válvula

En muchos sistemas prácticos, la constante de tiempo de la válvula es muy pequeña comparada con las constantes de tiempo de otros componentes del sistema de control, y la función de transferencia de la válvula puede ser aproximada a una constante.

$$G_V(s) = \frac{Q(s)}{U(s)} = K_V$$

Bajo estas condiciones, la válvula contribuye con un retardo dinámico despreciable.

Para la válvula BPX2-97A:

$$Cv = 9.28 \frac{gal}{min}$$

$$Kv = 8m^3/h$$

6.3.4. Modelo matemático del tanque:

En este caso lo expresaremos en función del tiempo con respecto a la altura del tanque:

$$\frac{dh}{dt} = \frac{Q_i - C_V . a_V \sqrt{2gh}}{A_t}$$

h_{max}: $altura\ del\ tanque\ = 60cm$

Q_i: $caudal\ de\ entada = 137.11\ l/min$

C_V: $coeficiente\ de\ la\ válvula = 9.28 \frac{gal}{min} = 8m^3/h$

$a_V = área\ de\ la\ válvula =$ ¾ ‘’=1.905cm

$A_t = área\ del\ tanque = 0.082957\ m^2$

6.3.5. Obtener la curva del caudal de la electroválvula

MUESTRA	ALTURA (cm)	TIEMPO 1 (s)	TIEMPO 2 (s)	TIEMPO 3 (s)	TIEMPO 4 (s)	TIEMPO PROM (s)	CAUDAL (m3/s)	CAUDAL PROM (m3/s)	CAUDAL PROM (l/min)
1365	10	11.18	10.96	11.56	12.1	11.45	0.000725	0.000758	45.50
	20	21.31	21.65	21.49	22.09	21.64	0.000767		
	30	32.38	32.51	32.87	33.69	32.86	0.000757		
	40	42.8	43.72	43	44.03	43.39	0.000765		
	50	52.32	53.11	54	53.79	53.29	0.000778		
2047	10	7.23	6.77	7.21	6.49	6.93	0.001198	0.001184	71.02
	20	13.27	13.41	14.22	13.34	13.56	0.001224		
	30	20.24	19.98	21.96	20.63	20.33	0.001207		
	40	27.5	27.06	27.68	27.22	27.37	0.001213		
	50	33.32	33.22	33.68	53.74	38.39	0.001078		
2730	10	5.11	4.57	7.2	4.5	5.35	0.001552	0.001653	99.21
	20	10.25	9.39	10.25	9.58	9.87	0.001681		
	30	16.71	14.77	16.71	14.92	15.78	0.001577		
	40	19.2	19.02	19.52	19.37	19.28	0.001721		
	50	23.71	23.57	24.76	23.57	23.9	0.001735		
6825	10	3.82	4.03	4.25	3.5	3.9	0.002127	0.002182	130.93
	20	7.41	7.94	7.83	7.41	7.65	0.002170		
	30	12.02	11.59	12.27	11.1	11.75	0.002119		
	40	14.74	14.79	14.77	14.96	14.82	0.002240		
	50	18.37	18.66	18.42	18.1	18.39	0.002256		
4095	10	4.49	4.03	5.78	4.3	4.65	0.001784	0.002285	137.11
	20	7.77	6.96	8.96	6.65	7.59	0.002187		
	30	10.28	9.95	12.21	9.85	10.57	0.002354		
	40	12.66	11.78	15.31	12.58	13.08	0.002536		
	50	15.54	15.08	18.45	15.63	16.18	0.002564		

Tabla 1. Cuadro elaborado en microsoft excel 2010

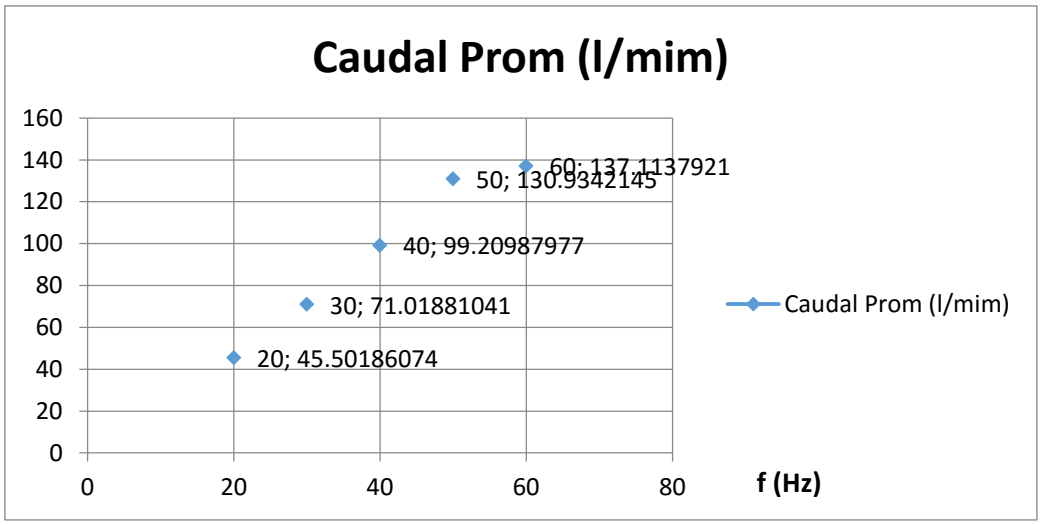


Figura 6.5. Curva del caudal de entrada de la electroválvula

6.4. DISEÑO Y CONSTRUCCIÓN DEL HARDWARE

Dentro de esta sección se describirá cada módulo que compone el Sistema de la Mini Planta de Control de Nivel, los cuales se mostrarán en esquemas eléctricos como en diseños PCB para poder apreciar el montaje de los distintos componentes.

6.4.1. Módulo para la Fuente de Voltaje de 5V

Para que los módulos de la Mini Planta estén respectivamente energizados, se ha diseñado una fuente de Voltaje fija de 5V, utilizando un regulador de voltaje LM7895, con su debidas protecciones ante cortocircuitos, los son los diodos rectificadores, como se muestra en la figura siguiente.

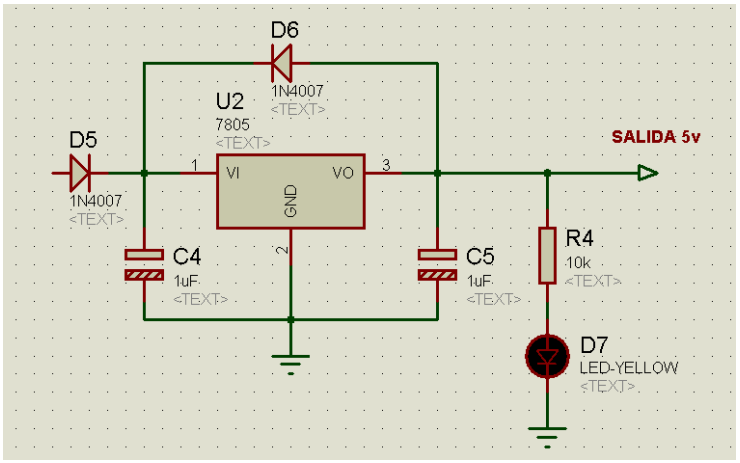


Figura 6.6. Circuito del Módulo de la Fuente de Voltaje de 5V

6.4.2. Módulo para Adaptación del LCD

Para hacer posible la comunicación, entre el conector header (espadín) macho de la placa STM32-V3 hacia el conector header hembra del Módulo Lcd, se tuvo que diseñar la siguiente placa para convertir el conector hembra del LCD a un conector macho, como se muestra en la figura siguiente.

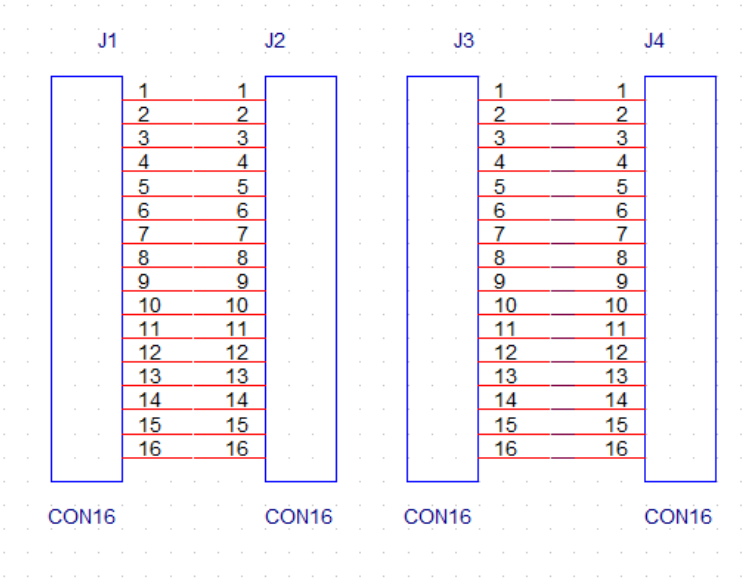


Figura 6.7. Circuito del Módulo de Adaptación del LCD

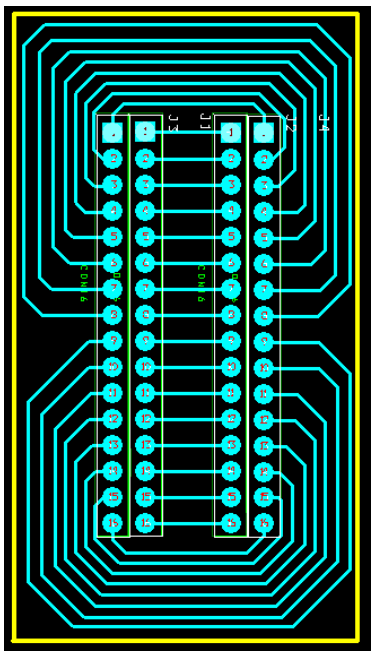


Figura 6.8. Diseño de PCB en Orcad de la Etapa de Adaptación del LCD

6.4.3. Módulo de Adaptación de la Placa STM32 hacia conectores de Salida

El Módulo que se muestra en la figura siguiente, representa el modulo principal de la Mini Planta, puesto que conecta la placa STM32-V3 hacia la interfaz del control de la electroválvula, los pulsadores y la interfaz del control de potencia de los pilotos.

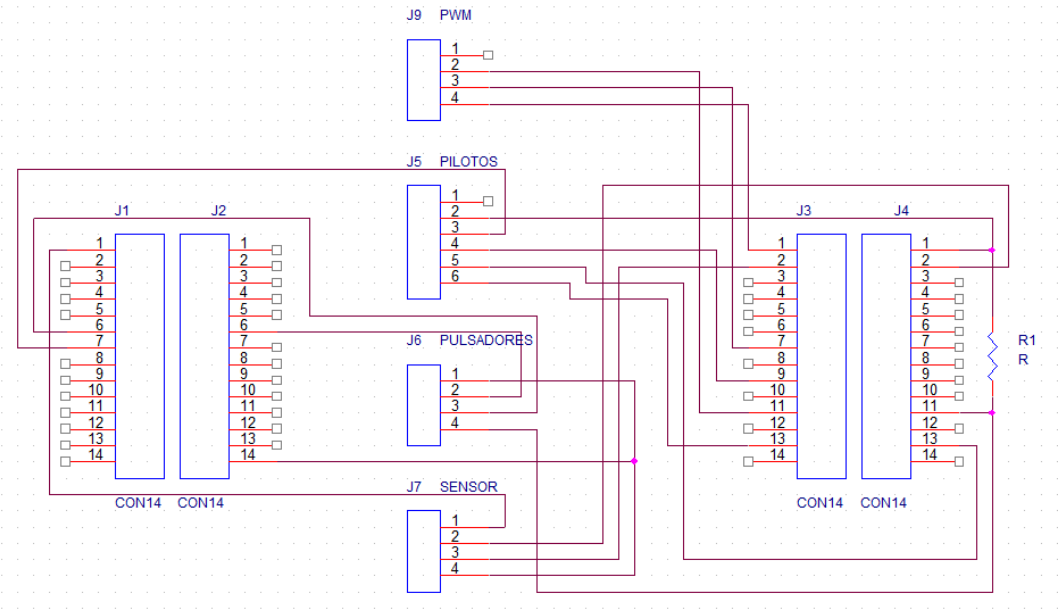


Figura 6.9. Circuito del Módulo de Adaptación de la Placa STM32 hacia conectores de Salida

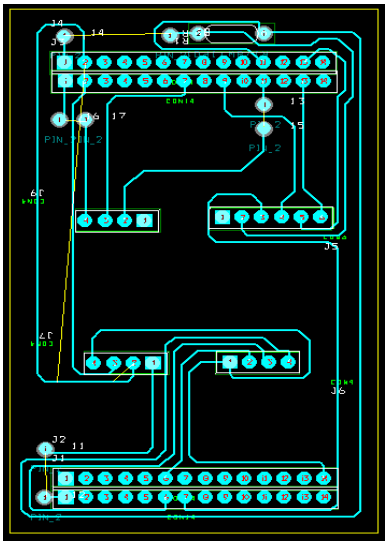


Figura 6.10. Diseño de PCB en Orcad de la Etapa de Adaptación de la Placa STM32 hacia conectores de Salida

6.4.4. Módulo para la Etapa de Potencia

El diseño a continuación tiene como objetivo dar la energía suficiente para el movimiento y activación de las cargar tales como electroválvulas y pilotos, dicho diseño se basa en la activación del triac BTA12, a través de un optoacoplador MOC3021, para que dicha activación solo sea a través de un pulso de 3.3V. Como se muestra en la figura siguiente.

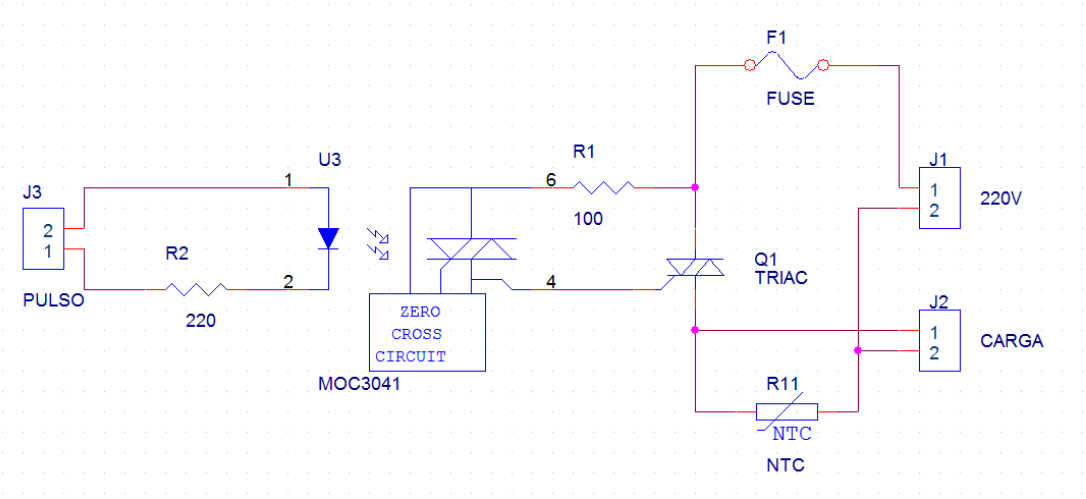


Figura 6.11. Circuito del Módulo de la Etapa de Potencia

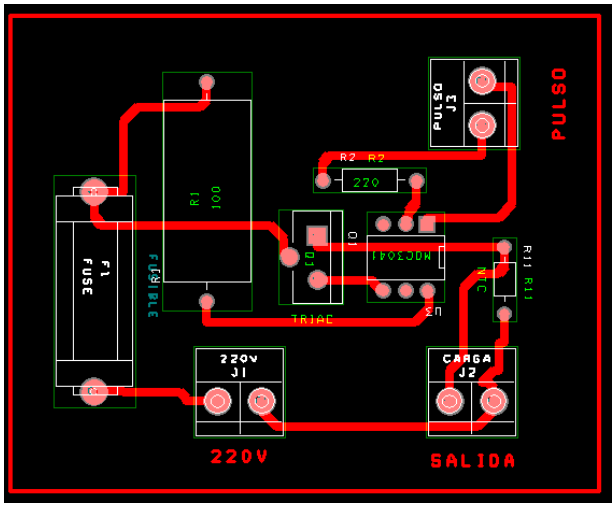


Figura 6.12. Diseño de PCB en Orcad de la Etapa de Potencia

6.4.5. Módulo para el Control de la Electroválvula

El diseño a continuación tiene objetivo captar un pulso de referencia, cada vez que la onda de 220V/60Hz pasa por cero. Para facilitar el proceso, se toma la onda de 220V y se reduce a través de un transformador a una señal de 6V, y se pasa a través de un puente de diodos rectificadores, para duplicar la frecuencia y hacer más fácil el proceso y quien es el responsable de enviar el pulso cada vez que la onda pasa por cero es el transistor 2N3906.

La onda de salida del transistor es deforme y se tiene que pasar a traves de un bufer, que este caso se utilizó una compuerta lógica 74HS04, para que la onda sea la adecuada.

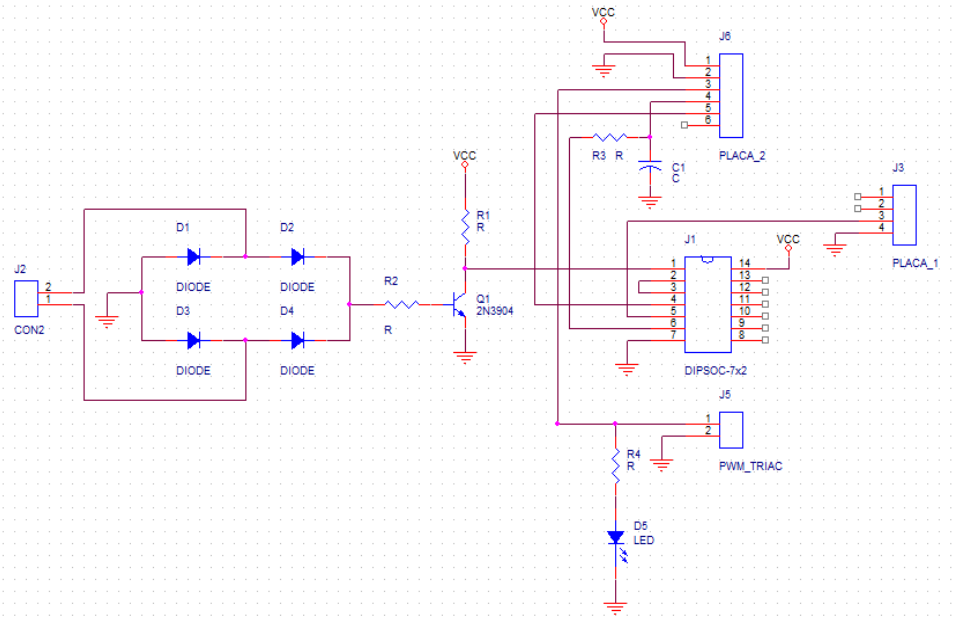


Figura 6.13. Circuito del Módulo para la Electroválvula

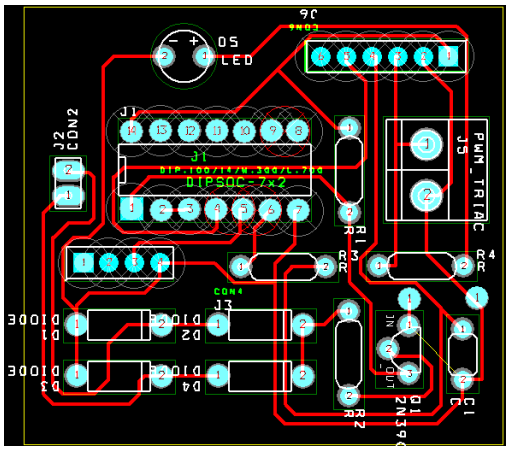


Figura 6.14. Diseño de PCB en Orcad de la Electroválvula

6.5. Desarrollo del Programa

En esta sección se describe el programa principal y las funciones que componen el software de la mini planta de nivel y cuyo código está escrito en lenguaje C y en el programan Keil uVision4.

6.5.1. Creación de un Nuevo Proyecto

Para iniciar con el programa para la mini planta de nivel, primero se debe de crear un nuevo proyecto en el software uVision4 y seguir los siguientes pasos como muestra la siguiente figura.

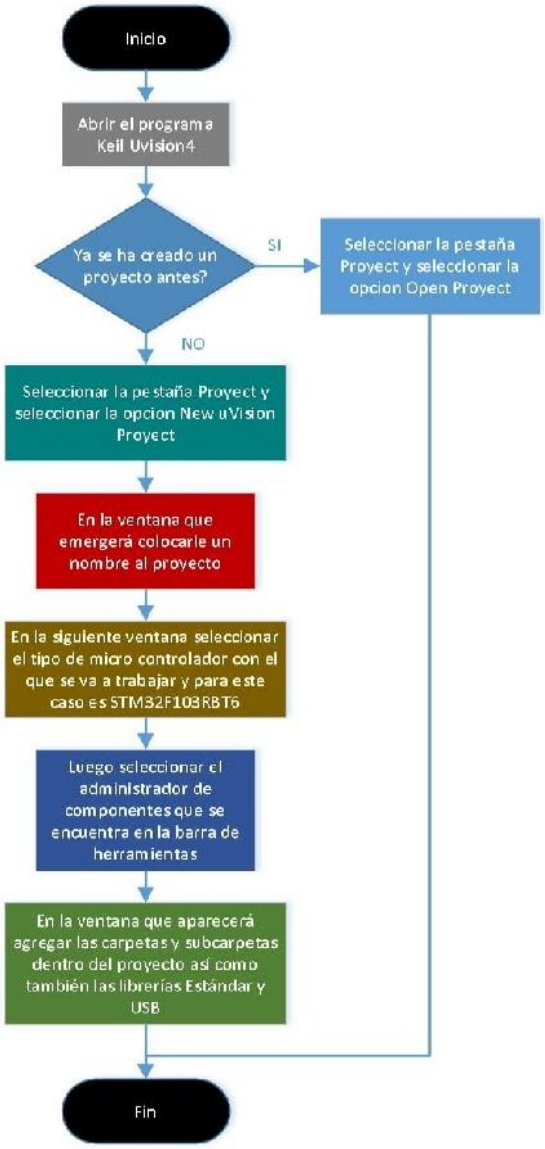


Figura 6.15. Diagrama de flujo para crear un nuevo proyecto

6.5.2. Configuración del Hardware

Para iniciar con el programa para la mini planta de nivel, se deben realizar las configuraciones de las funciones correspondientes, para la habilitación de cada módulo del hardware, llámese tal Configuración del reloj uCOS, Configuración de Entradas y Salidas, Configuración de Timer, Configuración de SPI, etc , como se muestra en la figura siguiente.

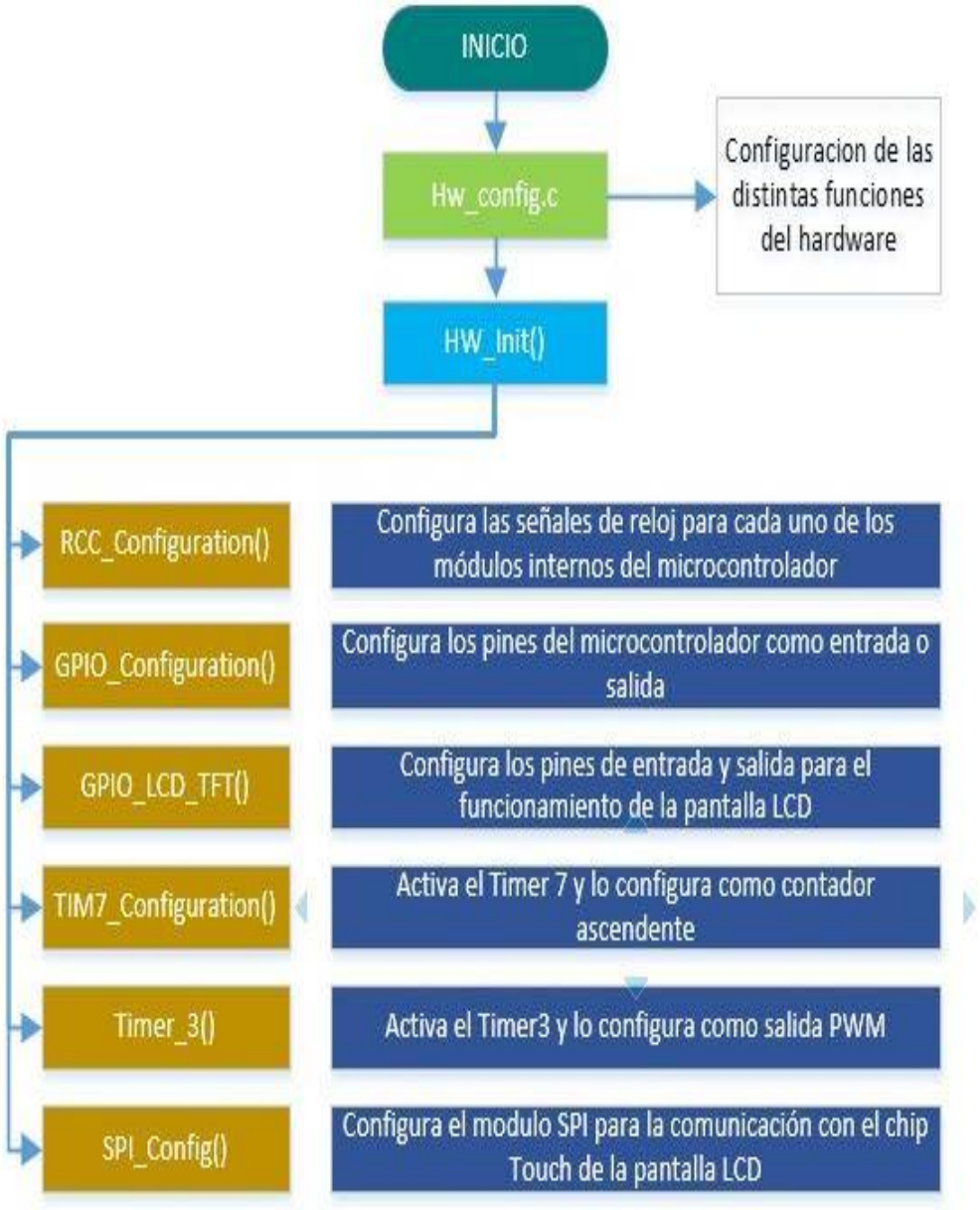


Figura 6.16. Configuración del Hardware

6.5.3. Configuración del Software

Para iniciar con el programa para la mini planta de nivel, se deben realizar las configuraciones de las funciones correspondientes, para habilitar cada función de los Sistemas en tiempo real, llámese tal Iniciar el Sistema, Habilitar el total de las funciones del Hardware descritas anteriormente, Iniciar con la Tarea Principal e Iniciar las Tareas Secundarias, tales como: AppTaskUserIF, AppTaskKbd, AppTaskAdc, AppTaskTim, como se muestra en la figura siguiente.

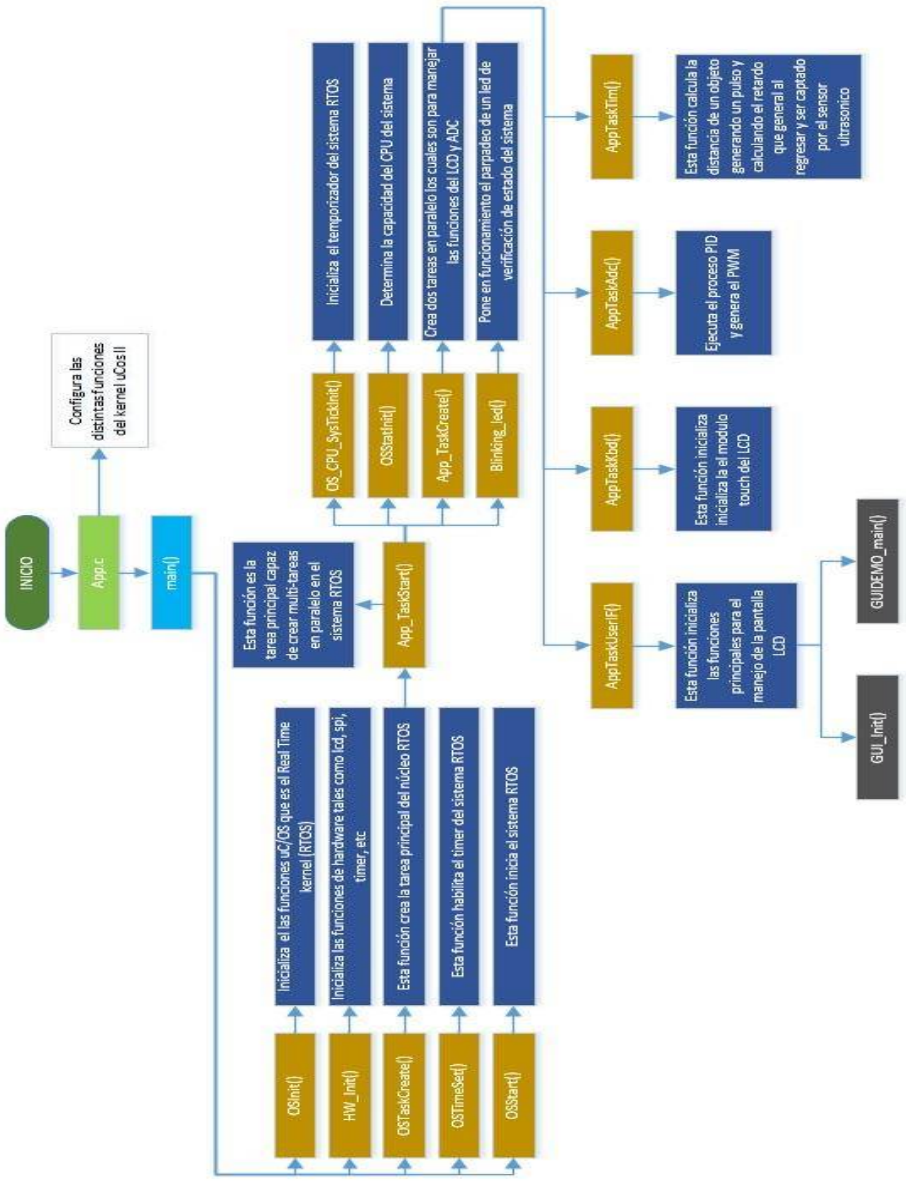


Figura 6.17. Configuración del Software

6.5.4. Distribución de las Librerías Standard

Para iniciar con el programa para la mini planta de nivel, se debe descargar de la página web STMicronics y añadir la Librería Estándar de Configuración de Módulos de Hardware. Dicha Librería Standard incluye los siguientes archivos, como muestra la siguiente figura.

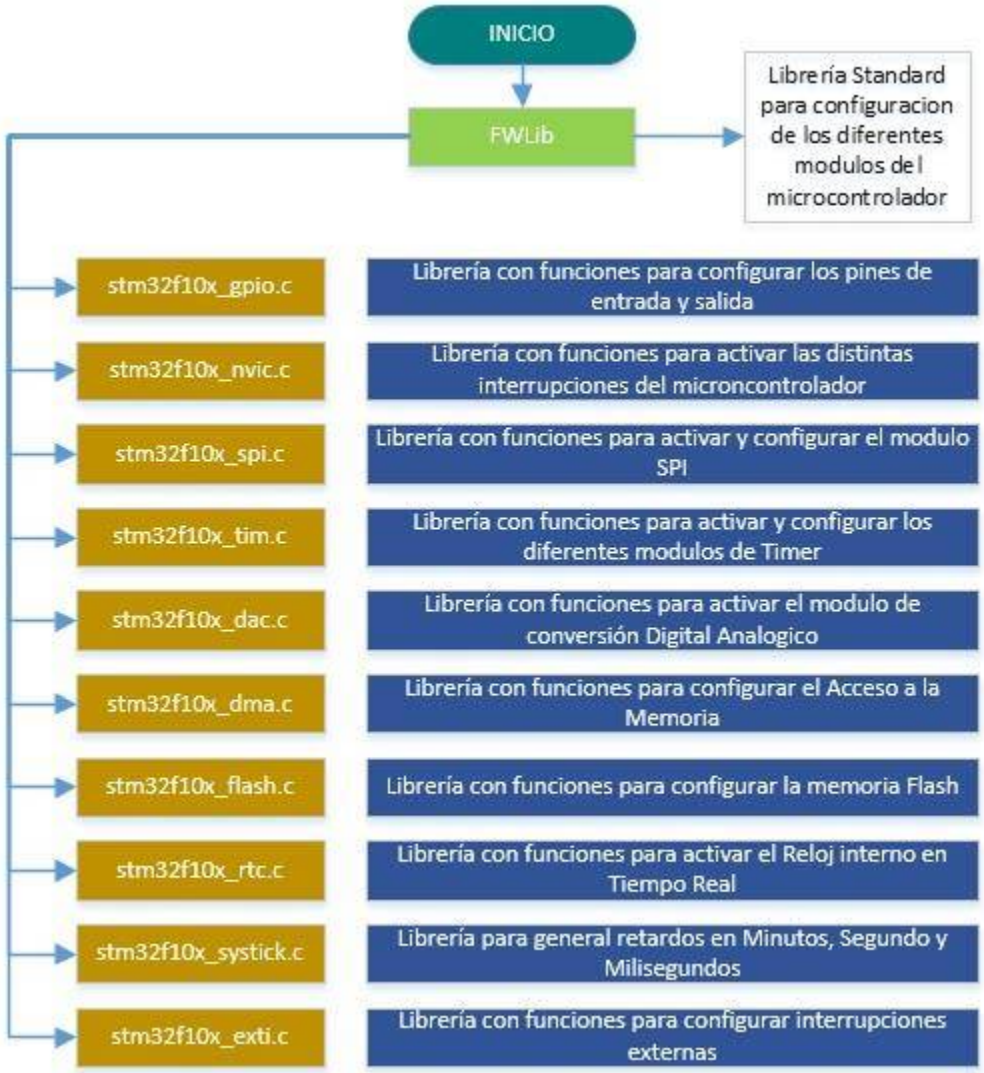


Figura 6.18. Distribución de las Librerías Standard

6.5.5. Descripción de las Tareas Principales del uCOS

Para continuar con el programa de la mini planta de nivel, a continuación de describe cada una de las Tareas Principales que se han configurado en el uCOS; AppTaskKbd, el cual habilita el módulo Touch; AppTaskAdc, el cual actualiza las variables controladas en el PID; AppTaskTim, quien se encarga de la activación del módulo timer, como muestra la siguiente figura.

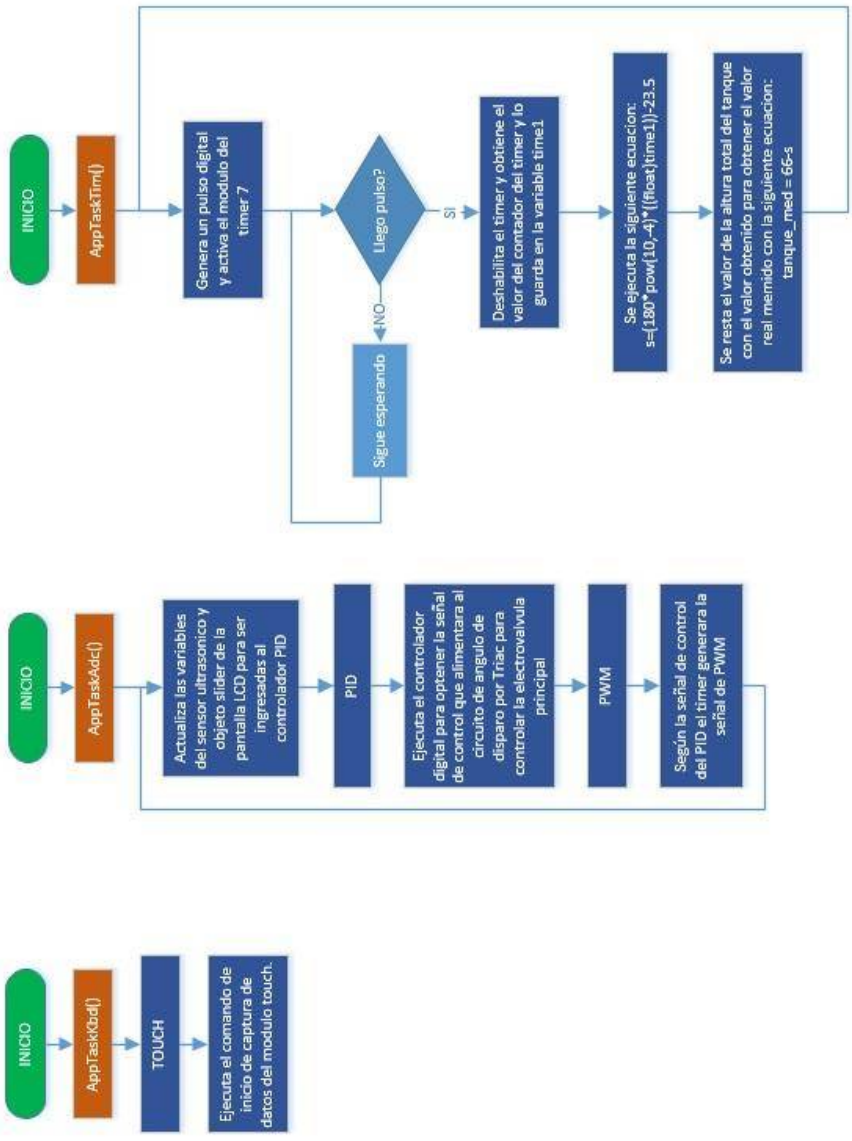


Figura 6.19. Descripción de las Tareas Principales del uCOS

6.5.6. Configuración del Programa Principal de la Visualización de la Pantalla

El programa Principal de la mini planta de nivel, configura las pantallas de control que se visualizan en el proyecto, así como la pantalla de visualización del estado de variables, como muestra la siguiente figura.

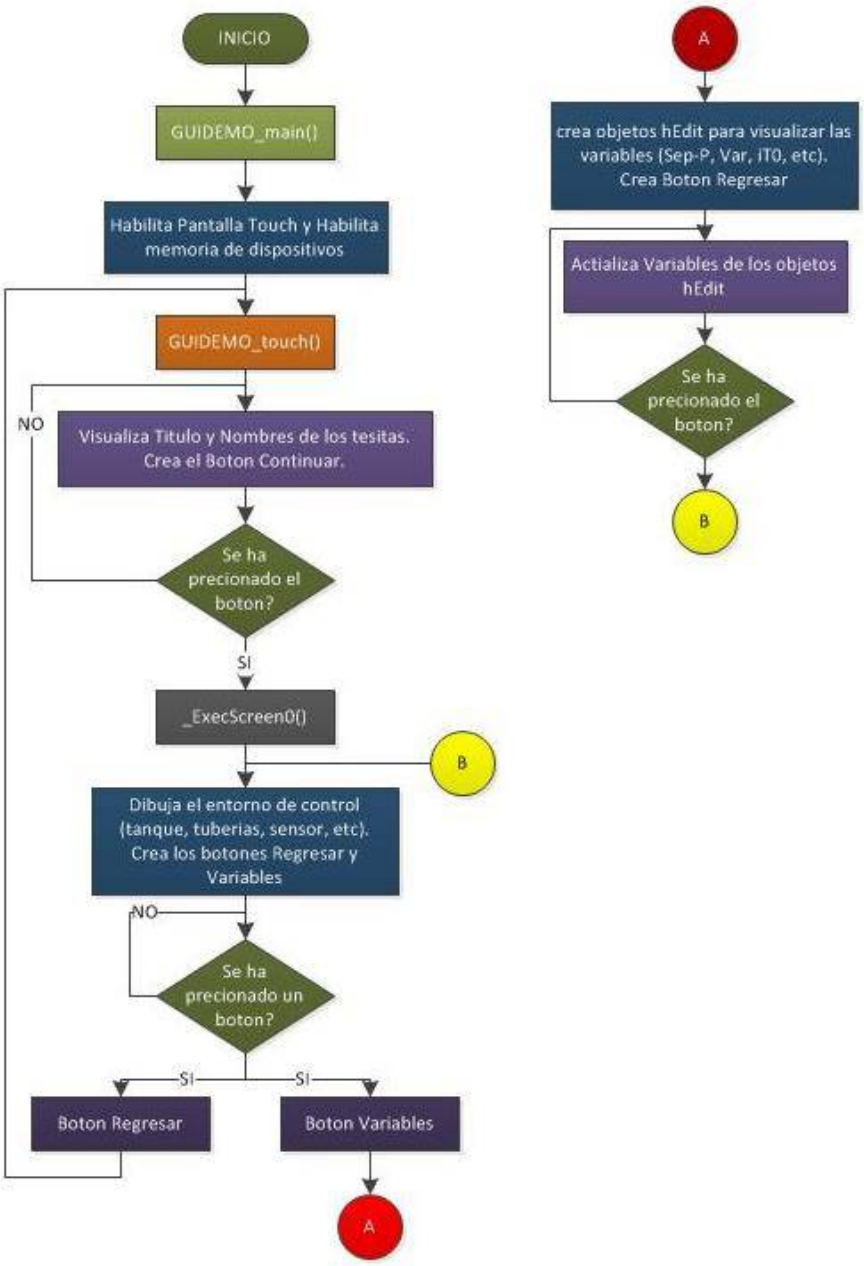


Figura 6.20. Configuración del Programa Principal de la Visualización de la pantalla

CAPÍTULO 7

Análisis de la Inversión del Proyecto

ESTRUCTURA					
ITEM	DETALLE	UNIDAD	CANTIDAD	V.U (S/.)	V.T (S/.)
1	Perfil Tubular Cuadrado Genérico 15x15 x 1.5 mm x 6 m	unidad	5	25.00	125.00
2	Plancha Galvanizada 0.25x1 m	metro	1	55.00	55.00
3	Polimetilmetacrilato	metro	1	50.00	50.00
4	Pegamento para Acrílico	litro	1/4	100.00	10.00
5	Bandejas Circulares	unidad	2	25.00	50.00
6	Ruedas Giratorias Pequeñas	unidad	4	2.50	10.00
7	Tubo Pvc 3/4"	unidad	1	15.00	15.00
8	Abrazadera para Tubería	unidad	2	2.00	4.00
9	Codos Pvc de 90° - 3/4"	unidad	2	1.50	3.00
10	Codos Pvc de 45° - 3/4"	unidad	1	1.00	1.00
11	Llave de Paso de Bola Pvc Lisa - 3/4"	unidad	1	5.00	5.00
12	Caja Metálica	unidad	2	35.00	70.00
13	Canaleta	unidad	2	2.00	4.00
14	Base Zincromato	galón	1	25.00	25.00
				TOTAL	427.00

Figura 7.1. Costo de la Estructura

FUENTE DE VOLTAJE DE 5V					
ITEM	DETALLE	UNIDAD	CANTIDAD	V.U (S/.)	V.T (S/.)
1	Lm7805	unidad	2	1.00	2.00
2	Conector para batería de 9V	unidad	2	0.50	1.00
3	Conector USB tipo A Hembra	unidad	2	1.00	2.00
4	Switch	unidad	2	0.50	1.00
5	Conector Molex de 02 pines	unidad	1	0.20	0.20
6	Diodo Rectificador 1N4007	unidad	6	0.10	0.60
7	Diodo Led	unidad	2	0.20	0.40
8	Placa Universal de 4.5mmx4.5mm	unidad	2	0.50	1.00
9	Resistencia	unidad	2	0.10	0.20
				TOTAL	8.40

Figura 7.2. Costo del Módulo de la Fuente de Voltaje de 5v

ADAPTACION DE LA PLACA STM32					
ITEM	DETALLE	UNIDAD	CANTIDAD	V.U (S/.)	V.T (S/.)
1	Fibra de Vidrio de 8.5mmx6mm	unidad	1	3.00	3.00
2	Placa de Evaluación STM32-V3	unidad	1	75.00	75.00
3	Conector Molex de 04 pines	unidad	3	0.50	1.50
4	Conector Molex de 06 pines	unidad	1	1.00	1.00
5	Espadín Hembra	unidad	1	2.00	2.00
6	Resistencia	unidad	1	0.10	0.10
				TOTAL	82.60

Figura 7.3. Costo del Módulo Adaptación de la Placa STM32 hacia conectores de Salida

ETAPA DE POTENCIA					
ITEM	DETALLE	UNIDAD	CANTIDAD	V.U (S/.)	V.T (S/.)
1	Triac BTA12	unidad	5	2.50	12.50
2	Resistencia de 100Ω / 5w	unidad	5	1.00	5.00
3	Resistencia de 220Ω/0.25w	unidad	5	0.10	0.50
4	MOC 3021	unidad	5	2.00	10.00
5	Porta Fusible para placa	unidad	5	0.80	4.00
6	Fusible de 3ª	unidad	5	0.30	1.50
7	Bornera de 02 pines	unidad	15	0.50	7.50
8	Fibra de Vidrio de 7.5mmx5.5mm	unidad	5	2.00	10.00
				TOTAL	51.00

Figura 7.4. Costo del Módulo para la Etapa de Potencia

CONTROL DE LA ELECTROVALVULA					
ITEM	DETALLE	UNIDAD	CANTIDAD	V.U (S/.)	V.T (S/.)
1	Fibra de Vidrio 4cmx4cm	unidad	1	1.50	1.50
2	Diodo Rectificador 1N4007	unidad	4	0.10	0.40
3	Transistor 2N3906	unidad	1	0.50	0.50
4	Compuerta Lógica 74HS04	unidad	1	1.00	1.00
5	Zócalo de 7x2	unidad	1	0.30	0.30
6	Resistencia	unidad	4	0.10	0.40
7	Conector Molex de 04 pines	unidad	1	0.50	0.50
8	Conector Molex de 06 pines	unidad	1	1.00	1.00
9	Bornera 02 pin	unidad	2	0.50	1.00
10	Condensador	unidad	1	0.20	0.20
				TOTAL	6.00

Figura 7.5. Costo del Módulo para el Control de la Electroválvula

ADAPTACION DEL LCD					
ITEM	DETALLE	UNIDAD	CANTIDAD	V.U (S/.)	V.T (S/.)
1	Fibra de Vidrio de 6.5mmx3.5mm	unidad	1	2.00	2.00
2	LCD TFT de 2.8”	unidad	1	40.00	40.00
3	Espadín Macho doble	unidad	1	2.00	2.00
4	Flag FC-34P	unidad	1	2.00	2.00
				TOTAL	46.00

Figura 7.6. Costo del Módulo de la Adaptación del LCD

OTROS GASTOS					
ITEM	DETALLE	UNIDAD	CANTIDAD	V.U (S/.)	V.T (S/.)
1	Carrete de Estaño	unidad	1	10.00	10.00
2	Cable mellizo 2 x16 #AWG Indeco	metro	6	1.70	10.20
3	Terminal	unidad	4	0.30	1.20
4	Cable GTP 18 #AWG (rojo y negro)	metro	10	0.50	5.00
5	Cable de conexión multi-filar	metro	5	0.20	1.00
6	Cable de conexión solido	metro	5	0.20	1.00
7	Perno de 0.5”	docena	3	1.00	3.00
8	Perno de 1”	docena	2	1.50	3.00
9	Perno de 1.5”	docena	2	2.00	4.00
10	Arandelas	docena	2	1.00	2.00
11	Silicona para Acrilico	unidad	1	10.00	10.00
				TOTAL	50.40

Figura 7.7. Costo Otros Gastos

COSTO TOTAL DEL PROYECTO		
ITEM	DETALLE	COSTO (S/.)
1	ESTRUCTURA	427.00
2	MODULO DE LA FUENTE DE VOLTAJE DE 5V	8.40
3	MODULO DE LA ADAPTACION DEL LCD	46.00
4	MODULO DE LA ADAPTACION DE LA PLACA STM32	82.60
5	MODULO PARA LA ETAPA DE POTENCIA	51.00
6	MODULO PARA EL CONTROL DE LA ELECTROVALVULA	6.00
7	OTROS	50.40
8	BIBLIOGRAFIA	300
9	HORAS HOMBRE (360 HORAS - 3 MESES)	6000
10	PASAJES Y OTROS SERVICIOS	300
11	IMPREVISTOS	200
	TOTAL	7471.4

Figura 7.8. Costo de la Inversión Total del Proyecto

CAPÍTULO 8

8. Conclusiones y Recomendaciones

8.1 Conclusiones:

- Se logró diseñar la miniplanta de control de nivel, tomando como referencia el modelo que existe en el laboratorio de ingeniería electrónica.
-

- Seleccionamos un sensor ultrasónico, una electroválvula , y en microcontrolador de 32 bits adecuado

- Logramos determinar el algoritmo matemático y así probar el funcionamiento del sistema de control obteniendo la toma de datos de prueba.

- Se verificó que los sistemas de control pueden ser tan eficientes en su funcionamiento como los sistemas implementados con controladores lógicos programables (PLC).

- Usando sistemas con microcontroladores avanzados la propuesta económica de construcción se reduce abismalmente en comparación con marcas reconocidas de controladores lógicos programables

8.2 Recomendaciones:

Una vez concluida la tesis, se considera interesante otros aspectos de mejora y se propone:

- Trabajar en la mejora de las placas electrónicas utilizando máquinas, herramientas basadas con tecnologías de Control Numérico Computarizado (CNC), el cual consiste en el uso de una computadora para controlar y monitorear los movimientos de una máquina herramienta.

- Trabajar con componentes de tecnología de montaje superficial, el cual permite la reducción de pesos y dimensiones en la fabricación de equipos.

- Trabajar en el cambio del control de fuerza, implementando un variador y retirando la electrobomba casera por una de mayor eficiencia y reconocida en el mercado.

Bibliografía

- Ogata, Katsuhiko. Ingeniería de Control Moderna. Controles PID e introducción al control robusto. Tercera edición. Página 669. Editorial Prentice Hall. 1998.

- Ogata, Katsuhiko. Sistemas de control en tiempo discreto. Segunda edición. Página 116. Editorial Prentice Hall. 1996

- Dylan Andres Alzate Rodríguez. Control y Medida de Nivel de Liquido con Señales de Ultrasonido. Colombia. 2010

- Víctor Napoleón López Medina. Diseño y Construcción del Modelo de una Planta de Nivel de Líquidos, Sistema Vasos Comunicantes. Ecuador. 2009

- Esteban Richmond Salazar. Diseño y Construcción de una Interfaz de Control de Nivel, Temperatura y Flujo de Agua en un Tanque para Uso en Prácticas de Laboratorio. Costa Rica. 2009

- José Carlos Garcés Pico, Juan Pablo León Calderón. Diseño e Implementación de una Planta de Nivel, Controlada mediante Redes Neuronales y Lógica Difusa, destinada al Laboratorio de Control Industrial de la Universidad de las Fuerzas Armadas Espe. Ecuador. Mayo 2015

- Ilber Adonayt Ruge Ruge. Método Básico para Implementar un Controlador Digital Pid en un Microcontrolador Pic para Desarrollo de Aplicaciones a Bajo Costo (Aplicaciones en Control de Potencia y la Industria). Universidad de Cundinamarca - Colombia

- Sandoval Ramos Alejandro. Sistema de Control de un Nivel aplicando un PID en un Modelo de Distribuidor de Colada Continua. México. Junio 2009

- Reference manual. STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced ARM-based 32-bit MCUs. Enero 2012

- ILI TECHNOLOGY CORP. a-Si TFT LCD Single Chip Driver 240RGBx320 Resolution and 262K color. Versión: V0.30. Taiwan

- UM0462 User manual. STM32™ and STM8™ Flash Loader Demonstrator. Doc ID 13916 Rev 7. Noviembre 2009

Anexos

PROGRAMA: GUIDEMO_TOUCH

```
/*
**********************************************************************
*                                uC/GUI
*                 Universal graphic software for embedded applications
*
*                 (c) Copyright 2002, Micrium Inc., Weston, FL
*                 (c) Copyright 2002, SEGGER Microcontroller Systeme GmbH
*
*           µC/GUI is protected by international copyright laws. Knowledge of the
*           source code may not be used to write a similar product. This file may
*           only be used in accordance with a license and should not be redistributed
*           in any way. We appreciate your understanding and fairness.
*
----------------------------------------------------------------------
File        : GUIDEMO_Touch
Purpose     : Touch demo
---------------------------END-OF-HEADER------------------------------
*/

#include "GUI.h"
#include "GUIDEMO.h"
#include "LCD_ConfDefaults.h"  /* valid LCD configuration */
#include "stm32f10x_adc.h"
#include "stm32f10x_gpio.h"

#if (GUI_WINSUPPORT && GUI_SUPPORT_TOUCH)

#include "..\GUIinc\BUTTON.h"
#include "..\GUIinc\EDIT.h"
#include "..\GUIinc\PROGBAR.h"
#include "..\GUIinc\SLIDER.h"

#define countof(Obj) (sizeof(Obj)/sizeof(Obj[0]))

#define PROGBAR_CF_HORIZONTAL (0 << 0)
#define PROGBAR_CF_VERTICAL   (1 << 0)

int _ExecKeyboard(void);
int _ExecScreen0(void);
int _ExecScreen1(void);


unsigned int CCR2_Val_0 = 4095, Set_point = 0;
extern GUI_CONST_STORAGE GUI_BITMAP bmTank_3;
extern GUI_CONST_STORAGE GUI_BITMAP bmred_button_not_pressed;
extern GUI_CONST_STORAGE GUI_BITMAP bmred_button_pressed;
extern GUI_CONST_STORAGE GUI_BITMAP bmgreen_button_not_pressed;
extern GUI_CONST_STORAGE GUI_BITMAP bmgreen_button_pressed;
extern GUI_CONST_STORAGE GUI_BITMAP bmYellow_pilot;
extern GUI_CONST_STORAGE GUI_BITMAP bmGreen_pilot;
```

```
extern GUI_CONST_STORAGE GUI_BITMAP bmRadar_level_transmitter;
extern GUI_CONST_STORAGE GUI_BITMAP bmCentrifugal_pump;
extern GUI_CONST_STORAGE GUI_BITMAP bmCentrifugal_pump_2;
extern GUI_CONST_STORAGE GUI_BITMAP bmRed_pushbutton;
extern GUI_CONST_STORAGE GUI_BITMAP bmh_pipe;
extern GUI_CONST_STORAGE GUI_BITMAP bmv_pipe;
extern GUI_CONST_STORAGE GUI_BITMAP bm90_curve;
extern GUI_CONST_STORAGE GUI_BITMAP bm90_curve_2;
extern GUI_CONST_STORAGE GUI_BITMAP bm90_curve_3;
extern GUI_CONST_STORAGE GUI_BITMAP bm90_curve_4;

extern int Flag_EN1, Flag_EN2, flag_Enganche, Flag_Emerg;
extern float s, tanque_med, tanque_med2;
extern u32 time1;
extern float uT,iT0,eT0;
/*********************************************************************
*       static data
**********************************************************************
*/
static const GUI_WIDGET_CREATE_INFO aDialogCreate[] = {
  { FRAMEWIN_CreateIndirect, "Set_P",                0,  128,  71, 40, 115, 0},
  { SLIDER_CreateIndirect,   NULL,     GUI_ID_SLIDER1, 10, 2, 15, 92, SLIDER_CF_VERTICAL, 0 },
};

/*********************************************************************
*       _cbCallback
**********************************************************************
*/
static void _cbCallback(WM_MESSAGE * pMsg) {
  //WM_HWIN hWin = pMsg->hWin;
  switch (pMsg->MsgId) {
    default:
      WM_DefaultProc(pMsg);
  }
}

/*********************************************************************
*            Exec Screen 0
**********************************************************************/
int _ExecScreen0(void) {

  #define ID_SCREEN_1  1
            #define ID_SCREEN_2   2
            #define ID_SCREEN_3   3
            #define ID_SCREEN_4   4
            #define ID_BACK_1       5
  int Key = 0;

  WM_HWIN hWin;
            BUTTON_Handle ahButton2[2];
            PROGBAR_Handle hProg;
            EDIT_Handle hEdit, hEdit2;

            GUI_SetBkColor(GUI_BK);
            GUI_Clear();
```

```
  hWin = GUI_CreateDialogBox(aDialogCreate, GUI_COUNTOF(aDialogCreate), _cbCallback, 0, 0, 0);
          SLIDER_SetRange(WM_GetDialogItem(hWin, GUI_ID_SLIDER1), 0, 4095);
          SLIDER_SetValue(WM_GetDialogItem(hWin, GUI_ID_SLIDER1), CCR2_Val_0);
          CCR2_Val_0 = SLIDER_GetValue(WM_GetDialogItem(hWin, GUI_ID_SLIDER1));
          Set_point = (55*CCR2_Val_0)/4095;

          ahButton2[0] =  BUTTON_Create( 125, 215, 70, 20, 9, BUTTON_CF_SHOW );
          ahButton2[1] =  BUTTON_Create( 12, 15, 70, 20, 10, BUTTON_CF_SHOW );
  BUTTON_SetText (ahButton2[0], "REGRESAR");
          BUTTON_SetBkColor(ahButton2[0], 0, GUI_YELLOW);
          BUTTON_SetFont(ahButton2[0], &GUI_Font13_ASCII);

  BUTTON_SetText (ahButton2[1], "VARIABLES");
          BUTTON_SetBkColor(ahButton2[1], 0, GUI_LIGHTRED);
          BUTTON_SetFont(ahButton2[1], &GUI_Font13_ASCII);

          hProg = PROGBAR_CreateEx(269, 81, 15, 92, 0, WM_CF_SHOW | WM_CF_STAYONTOP|
WM_CF_MEMDEV, PROGBAR_CF_VERTICAL, 0);
          PROGBAR_SetBarColor(hProg, 0, GUI_BLUE);
          PROGBAR_SetBarColor(hProg, 1, GUI_BK);
          PROGBAR_SetTextColor(hProg, 1, GUI_BLACK);
          PROGBAR_SetTextColor(hProg, 0, GUI_LIGHTYELLOW);
          PROGBAR_SetMinMax(hProg, 0, 60);

          hEdit = EDIT_CreateEx(128, 49, 40, 15, 0,  WM_CF_SHOW | WM_CF_STAYONTOP | WM_CF_MEMDEV,
0, 0, 5);
          EDIT_SetFloatMode(hEdit, 55-Set_point, 0, 100, 2, 0);
          hEdit2 = EDIT_CreateEx(220, 35, 40, 15, 0,  WM_CF_SHOW | WM_CF_STAYONTOP |
WM_CF_MEMDEV, 0, 0, 5);
          EDIT_SetFloatMode(hEdit2, 0, 0, 100, 2, 0);

          GUI_SetFont(&GUI_Font13B_ASCII);
  GUI_SetTextMode(GUI_TM_TRANS);
  GUI_DispStringHCenterAt("CONTROL DE NIVEL", 160, 10);
  GUI_DispStringHCenterAt("DE AGUA", 160, 25);
  GUI_SetColor(GUI_BK);
  GUI_FillRect(12, 40, 102, 160);
  GUI_FillRect(12, 169, 102, 229);
  GUI_SetColor(GUI_WHITE);
  GUI_DispStringAt("PILOTO", 41, 88);
  GUI_DispStringAt("STOP", 22, 147);
  GUI_DispStringAt("START", 64, 147);
  GUI_DispStringAt("P. EMERG.", 33, 214);

  GUI_DrawBitmapEx(&bmYellow_pilot, 38, 43, 0, 0, 1000, 1000);
  GUI_DrawBitmapEx(&bmred_button_not_pressed, 16, 102, 0, 0, 1000, 1000);
  GUI_DrawBitmapEx(&bmgreen_button_not_pressed, 59, 102, 0, 0, 1000, 1000);
  GUI_DrawBitmapEx(&bmRed_pushbutton, 38, 171, 0, 0, 1000, 1000);

  GUI_DrawBitmapEx(&bm90_curve, 193, 59, 0, 0, 1000, 1000);
  GUI_DrawBitmapEx(&bm90_curve_2, 231, 59, 0, 0, 1000, 1000);
  GUI_DrawBitmapEx(&bm90_curve_2, 294, 156, 0, 0, 1000, 1000);
  GUI_DrawBitmapEx(&bm90_curve_3, 174, 215, 0, 0, 1000, 1000);
  GUI_DrawBitmapEx(&bm90_curve_4, 294, 215, 0, 0, 1000, 1000);

  GUI_DrawBitmapEx(&bmv_pipe, 193, 71, 0, 0, 1000, 3314);
```

```
GUI_DrawBitmapEx(&bmv_pipe, 303, 169, 0, 0, 1000, 1342);
GUI_DrawBitmapEx(&bmh_pipe, 206, 59, 0, 0, 1000, 1000);
GUI_DrawBitmapEx(&bmh_pipe, 187, 224, 0, 0, 3264, 1000);

GUI_DrawBitmapEx(&bmTank_3, 215, 71, 0, 0, 1000, 1000);
GUI_DrawBitmapEx(&bmCentrifugal_pump, 151, 183, 0, 0, 1000, 1000);
        GUI_DrawBitmapEx(&bmRadar_level_transmitter, 264, 18, 0, 0, 1000, 1000);

/* Handle Keyboard until ESC or ENTER is pressed */
do {
                    if(Flag_Emerg == 1)
                    {
                            if(Flag_EN1 == 1)
                            {
                                    GUI_DrawBitmapEx(&bmgreen_button_pressed, 59, 102, 0, 0, 1000, 1000);
                            }
                            else
                            {
                                    GUI_DrawBitmapEx(&bmgreen_button_not_pressed, 59, 102, 0, 0, 1000, 1000);
                            }

                            if(Flag_EN2 == 1)
                            {
                                    GUI_DrawBitmapEx(&bmred_button_pressed, 16, 102, 0, 0, 1000, 1000);
                            }
                            else
                            {
                                    GUI_DrawBitmapEx(&bmred_button_not_pressed, 16, 102, 0, 0, 1000, 1000);
                            }

                            if(flag_Enganche == 1)
                            {
                                    GUI_DrawBitmapEx(&bmGreen_pilot, 38, 43, 0, 0, 1000, 1000);
                                    GUI_DrawBitmapEx(&bmCentrifugal_pump_2, 151, 183, 0, 0, 1000, 1000);
                                    CCR2_Val_0 = SLIDER_GetValue(WM_GetDialogItem(hWin,GUI_ID_SLIDER1));
                                    Set_point = (55*CCR2_Val_0)/4095;
                                    EDIT_SetFloatMode(hEdit, 55-Set_point, 0, 100, 2, 0);
                                    EDIT_SetFloatMode(hEdit2, tanque_med2, 0, 100, 2, 0);
                                    PROGBAR_SetValue(hProg, tanque_med2);
                            }
                            else
                            {
                                    GUI_DrawBitmapEx(&bmYellow_pilot, 38, 43, 0, 0, 1000, 1000);
                                    GUI_DrawBitmapEx(&bmCentrifugal_pump, 151, 183, 0, 0, 1000, 1000);
                            }
                    }
                    else
                    {
                            GUI_DrawBitmapEx(&bmgreen_button_not_pressed, 59, 102, 0, 0, 1000, 1000);
                            GUI_DrawBitmapEx(&bmred_button_not_pressed, 16, 102, 0, 0, 1000, 1000);
                            GUI_DrawBitmapEx(&bmYellow_pilot, 38, 43, 0, 0, 1000, 1000);
                            GUI_DrawBitmapEx(&bmCentrifugal_pump, 151, 183, 0, 0, 1000, 1000);
                    }
                    GUI_Delay(20);
                    if(BUTTON_IsPressed(ahButton2[0]))
                    {
```

```
                Key = 9;
            }
            if(BUTTON_IsPressed(ahButton2[1]))
            {
                BUTTON_Delete(ahButton2[0]);
                BUTTON_Delete(ahButton2[1]);
                EDIT_Delete(hEdit);
                EDIT_Delete(hEdit2);
                PROGBAR_Delete(hProg);
                if (WM_IsWindow(hWin)) {
                        WM_DeleteWindow(hWin);
                }
                GUI_SetBkColor(GUI_BK);
                GUI_SetColor(GUI_BLACK);
                GUI_Clear();

                _ExecScreen1();

                hWin = GUI_CreateDialogBox(aDialogCreate, GUI_COUNTOF(aDialogCreate),
_cbCallback, 0, 0, 0);
                SLIDER_SetRange(WM_GetDialogItem(hWin, GUI_ID_SLIDER1), 0, 4095);
                SLIDER_SetValue(WM_GetDialogItem(hWin, GUI_ID_SLIDER1), CCR2_Val_0);
                CCR2_Val_0 = SLIDER_GetValue(WM_GetDialogItem(hWin, GUI_ID_SLIDER1));
                Set_point = (55*CCR2_Val_0)/4095;

                ahButton2[0] =  BUTTON_Create( 125, 215, 70, 20, 9, BUTTON_CF_SHOW );
                ahButton2[1] =  BUTTON_Create( 12, 15, 70, 20, 10, BUTTON_CF_SHOW );
                BUTTON_SetText (ahButton2[0], "REGRESAR");
                BUTTON_SetBkColor(ahButton2[0], 0, GUI_YELLOW);
                BUTTON_SetFont(ahButton2[0], &GUI_Font13_ASCII);

                BUTTON_SetText (ahButton2[1], "VARIABLES");
                BUTTON_SetBkColor(ahButton2[1], 0, GUI_LIGHTRED);
                BUTTON_SetFont(ahButton2[1], &GUI_Font13_ASCII);

                hProg = PROGBAR_CreateEx(269, 81, 15, 92, 0, WM_CF_SHOW | WM_CF_STAYONTOP|
WM_CF_MEMDEV, PROGBAR_CF_VERTICAL, 0);
                PROGBAR_SetBarColor(hProg, 0, GUI_BLUE);
                PROGBAR_SetBarColor(hProg, 1, GUI_BK);
                PROGBAR_SetTextColor(hProg, 1, GUI_BLACK);
                PROGBAR_SetTextColor(hProg, 0, GUI_LIGHTYELLOW);
                PROGBAR_SetMinMax(hProg, 0, 60);

                hEdit = EDIT_CreateEx(128, 49, 40, 15, 0,  WM_CF_SHOW | WM_CF_STAYONTOP |
WM_CF_MEMDEV, 0, 0, 5);
                EDIT_SetFloatMode(hEdit, 55-Set_point, 0, 100, 2, 0);
                hEdit2 = EDIT_CreateEx(220, 35, 40, 15, 0,  WM_CF_SHOW | WM_CF_STAYONTOP |
WM_CF_MEMDEV, 0, 0, 5);
                EDIT_SetFloatMode(hEdit2, 0, 0, 100, 2, 0);

                GUI_SetFont(&GUI_Font13B_ASCII);
                GUI_SetTextMode(GUI_TM_TRANS);
                GUI_DispStringHCenterAt("CONTROL DE NIVEL", 160, 10);
                GUI_DispStringHCenterAt("DE AGUA", 160, 25);
                GUI_SetColor(GUI_BK);
                GUI_FillRect(12, 40, 102, 160);
```

```
                        GUI_FillRect(12, 169, 102, 229);
                        GUI_SetColor(GUI_WHITE);
                        GUI_DispStringAt("PILOTO", 41, 88);
                        GUI_DispStringAt("STOP", 22, 147);
                        GUI_DispStringAt("START", 64, 147);
                        GUI_DispStringAt("P. EMERG.", 33, 214);

                        GUI_DrawBitmapEx(&bmYellow_pilot, 38, 43, 0, 0, 1000, 1000);
                        GUI_DrawBitmapEx(&bmred_button_not_pressed, 16, 102, 0, 0, 1000, 1000);
                        GUI_DrawBitmapEx(&bmgreen_button_not_pressed, 59, 102, 0, 0, 1000, 1000);
                        GUI_DrawBitmapEx(&bmRed_pushbutton, 38, 171, 0, 0, 1000, 1000);

                        GUI_DrawBitmapEx(&bm90_curve, 193, 59, 0, 0, 1000, 1000);
                        GUI_DrawBitmapEx(&bm90_curve_2, 231, 59, 0, 0, 1000, 1000);
                        GUI_DrawBitmapEx(&bm90_curve_2, 294, 156, 0, 0, 1000, 1000);
                        GUI_DrawBitmapEx(&bm90_curve_3, 174, 215, 0, 0, 1000, 1000);
                        GUI_DrawBitmapEx(&bm90_curve_4, 294, 215, 0, 0, 1000, 1000);

                        GUI_DrawBitmapEx(&bmv_pipe, 193, 71, 0, 0, 1000, 3314);
                        GUI_DrawBitmapEx(&bmv_pipe, 303, 169, 0, 0, 1000, 1342);
                        GUI_DrawBitmapEx(&bmh_pipe, 206, 59, 0, 0, 1000, 1000);
                        GUI_DrawBitmapEx(&bmh_pipe, 187, 224, 0, 0, 3264, 1000);

                        GUI_DrawBitmapEx(&bmTank_3, 215, 71, 0, 0, 1000, 1000);
                        GUI_DrawBitmapEx(&bmCentrifugal_pump, 151, 183, 0, 0, 1000, 1000);
                        GUI_DrawBitmapEx(&bmRadar_level_transmitter, 264, 18, 0, 0, 1000, 1000);
                }
 } while (Key != 9);

        BUTTON_Delete(ahButton2[0]);
        BUTTON_Delete(ahButton2[1]);
        EDIT_Delete(hEdit);
        EDIT_Delete(hEdit2);
        PROGBAR_Delete(hProg);

 if (WM_IsWindow(hWin)) {
   WM_DeleteWindow(hWin);
 }
        GUI_SetBkColor(GUI_BK);
        GUI_Clear();
 return Key;
}

/*********************************************************************
*           Exec Screen 1
**********************************************************************/
int _ExecScreen1(void) {
        int Key;

        BUTTON_Handle ahButton3[1];
        EDIT_Handle hEdit3[6];

        GUI_RECT rText6, rText7, rText8, rText9, rText10, rText11, rText12, rText13;/*= {0, 80, XSize, 120};*/

        rText6.x0=30;
        rText6.y0=50;
```

```
rText6.x1=180;
rText6.y1=70;

rText7.x0=45;
rText7.y0=70;
rText7.x1=100;
rText7.y1=90;

rText8.x0=45;
rText8.y0=90;
rText8.x1=100;
rText8.y1=110;

rText9.x0=45;
rText9.y0=110;
rText9.x1=100;
rText9.y1=130;

rText10.x0=30;
rText10.y0=130;
rText10.x1=180;
rText10.y1=150;

rText11.x0=45;
rText11.y0=150;
rText11.x1=100;
rText11.y1=170;

rText12.x0=45;
rText12.y0=170;
rText12.x1=100;
rText12.y1=190;

rText13.x0=45;
rText13.y0=190;
rText13.x1=100;
rText13.y1=210;

GUI_SetFont(&GUI_Font16B_ASCII);
GUI_SetTextMode(GUI_TM_TRANS);
GUI_DispStringHCenterAt("VARIABLES", 160, 10);
GUI_SetFont(&GUI_Font16_ASCII);
GUI_DispStringInRect("Variables de Proceso :", &rText6, GUI_TA_LEFT | GUI_TA_VCENTER);
GUI_SetFont(&GUI_Font16B_ASCII);
GUI_DispStringInRect("Set-P :", &rText7, GUI_TA_LEFT | GUI_TA_VCENTER);
GUI_DispStringInRect("Var :", &rText8, GUI_TA_LEFT | GUI_TA_VCENTER);
GUI_DispStringInRect("Valvula :", &rText9, GUI_TA_LEFT | GUI_TA_VCENTER);

GUI_SetFont(&GUI_Font16_ASCII);
GUI_DispStringInRect("Variables de PID :", &rText10, GUI_TA_LEFT | GUI_TA_VCENTER);
GUI_SetFont(&GUI_Font16B_ASCII);
GUI_DispStringInRect("iT0 :", &rText11, GUI_TA_LEFT | GUI_TA_VCENTER);
GUI_DispStringInRect("eT0 :", &rText12, GUI_TA_LEFT | GUI_TA_VCENTER);
GUI_DispStringInRect("uT :", &rText13, GUI_TA_LEFT | GUI_TA_VCENTER);

ahButton3[0] =  BUTTON_Create( 125, 215, 70, 20, 31, BUTTON_CF_SHOW );
```

```
 BUTTON_SetText (ahButton3[0], "REGRESAR");
        BUTTON_SetBkColor(ahButton3[0], 0, GUI_YELLOW);
        BUTTON_SetFont(ahButton3[0], &GUI_Font13_ASCII);

        hEdit3[0] = EDIT_CreateEx(110, 70, 40, 15, 0,  WM_CF_SHOW | WM_CF_STAYONTOP |
WM_CF_MEMDEV, 0, 0, 5);
        EDIT_SetFloatMode(hEdit3[0], 55-Set_point, 0, 100, 2, 0);
        hEdit3[1] = EDIT_CreateEx(110, 90, 40, 15, 0,  WM_CF_SHOW | WM_CF_STAYONTOP |
WM_CF_MEMDEV, 0, 0, 5);
        EDIT_SetFloatMode(hEdit3[1], tanque_med2, 0, 100, 2, 0);

        hEdit3[2] = EDIT_CreateEx(110, 110, 40, 15, 0,  WM_CF_SHOW | WM_CF_STAYONTOP |
WM_CF_MEMDEV, 0, 0, 5);
        EDIT_SetTextAlign(hEdit3[2], GUI_TA_HCENTER | GUI_TA_VCENTER);
        EDIT_SetText(hEdit3[2], "OFF");

        hEdit3[3] = EDIT_CreateEx(110, 150, 40, 15, 0,  WM_CF_SHOW | WM_CF_STAYONTOP |
WM_CF_MEMDEV, 0, 0, 5);
        EDIT_SetFloatMode(hEdit3[3], iT0, 0, 100, 1, 0);
        hEdit3[4] = EDIT_CreateEx(110, 170, 40, 15, 0,  WM_CF_SHOW | WM_CF_STAYONTOP |
WM_CF_MEMDEV, 0, 0, 5);
        EDIT_SetFloatMode(hEdit3[4], eT0, -100, 100, 1, 0);
        hEdit3[5] = EDIT_CreateEx(110, 190, 40, 15, 0,  WM_CF_SHOW | WM_CF_STAYONTOP |
WM_CF_MEMDEV, 0, 0, 5);
        EDIT_SetFloatMode(hEdit3[5], uT, 0, 100, 1, 0);

 do {
                EDIT_SetFloatValue(hEdit3[0], 55-Set_point);
                EDIT_SetFloatValue(hEdit3[1], tanque_med2);
                EDIT_SetText(hEdit3[2], "OFF");
                EDIT_SetFloatValue(hEdit3[3], iT0);
                EDIT_SetFloatValue(hEdit3[4], eT0);
                EDIT_SetFloatValue(hEdit3[5], uT);
                GUI_Delay(200);
                if(BUTTON_IsPressed(ahButton3[0]))
                {
                        Key = 31;
                }
 } while (Key != 31);

        BUTTON_Delete(ahButton3[0]);
        EDIT_Delete(hEdit3[0]);
        EDIT_Delete(hEdit3[1]);
        EDIT_Delete(hEdit3[2]);
        EDIT_Delete(hEdit3[3]);
        EDIT_Delete(hEdit3[4]);
        EDIT_Delete(hEdit3[5]);

        GUI_SetBkColor(GUI_BK);
        GUI_Clear();
 return Key;
}


/*********************************************************************
*           USER_Task
```

```
**********************************************************************/

#if GUIDEMO_LARGE

void GUIDEMO_Touch(void) {

  #define ID_SCREEN_0  1
  int i, r = 0;
  int XSize = LCD_GetXSize();
  int YSize = LCD_GetYSize();
  int XMid = XSize / 2;
  int YMid = YSize / 2;

        GUI_RECT rText, rText2, rText3, rText4, rText5;/*= {0, 80, XSize, 120};*/
        rText.x0=0;
        rText.y0=30;
        rText.x1=XSize;
        rText.y1=50;

        rText2.x0=0;
        rText2.y0=50;
        rText2.x1=XSize;
        rText2.y1=70;

        rText3.x0=30;
        rText3.y0=110;
        rText3.x1=100;
        rText3.y1=130;

        rText4.x0=30;
        rText4.y0=130;
        rText4.x1=160;
        rText4.y1=150;

        rText5.x0=30;
        rText5.y0=150;
        rText5.x1=160;
        rText5.y1=170;

        GUI_SetBkColor(GUI_BK);
        GUI_Clear();
  do {
                BUTTON_Handle ahButton[2];
                GUI_SetFont(&GUI_Font16B_ASCII);
                GUI_SetBkColor(GUI_YELLOW);
                GUI_SetColor(GUI_BLACK);
                GUI_SetTextMode(GUI_TM_NORMAL);
                GUI_DispStringInRect("CONTROL DE NIVEL DE AGUA", &rText, GUI_TA_HCENTER |
GUI_TA_VCENTER);
                GUI_DispStringInRect("USANDO MICROCONTROLADORES 32 BITS", &rText2,
GUI_TA_HCENTER | GUI_TA_VCENTER);

                GUI_SetFont(&GUI_Font16_ASCII);
                GUI_SetTextMode(GUI_TM_TRANS);
                GUI_DispStringInRect("Autores:", &rText3, GUI_TA_LEFT | GUI_TA_VCENTER);
                GUI_SetFont(&GUI_Font16B_ASCII);
```

```
                GUI_DispStringInRect("Bach. Bacheli A.", &rText4, GUI_TA_LEFT | GUI_TA_VCENTER);
                GUI_DispStringInRect("Bach. Fernandez R.", &rText5, GUI_TA_LEFT | GUI_TA_VCENTER);

  ahButton[0] =  BUTTON_Create( XMid - 35, YMid + 95, 70, 20, ID_SCREEN_0, BUTTON_CF_SHOW );
  BUTTON_SetText (ahButton[0], "CONTINUAR");
                BUTTON_SetBkColor(ahButton[0], 0, GUI_YELLOW);
                BUTTON_SetFont(ahButton[0], &GUI_Font13_ASCII);
  //r = GUI_GetKey();

                #if GUI_WINSUPPORT
                        WM_ExecIdle();
                #endif

                if(BUTTON_IsPressed(ahButton[0]))
                {
                        r = ID_SCREEN_0;
                }

                for (i=0; i< countof(ahButton); i++) {
                        BUTTON_Delete(ahButton[i]);
                }
  switch (r) {
                        case ID_SCREEN_0:
                        {
                                r = 0;
                                _ExecScreen0();
                                break;
                        }
  }
 } while ((r!='n') && (r!='N'));
}

#endif

#endif /* #if GUI_WINSUPPORT */
```

PROGRAMA: BSP.C

```
/*
*********************************************************************************************************
*                                       MICIRUM BOARD SUPPORT PACKAGE
*
*                                  (c) Copyright 2007; Micrium, Inc.; Weston, FL
*                         All rights reserved.  Protected by international copyright laws.
*                         Knowledge of the source code may NOT be used to develop a similar product.
*                         Please help us continue to provide the Embedded community with the finest
*                         software available.  Your honesty is greatly appreciated.
*********************************************************************************************************
*/

/*
*********************************************************************************************************
*                                       BOARD SUPPORT PACKAGE
```

```
*                          ST Microelectronics STM32
*                                with the
*                         STM3210B-EVAL Evaluation Board
* Filename      : bsp.c
* Version       : V1.00
* Programmer(s) : Brian Nagel
*********************************************************************************************************
*/

/*
*********************************************************************************************************
*                                        INCLUDE FILES
*********************************************************************************************************
*/

#define  BSP_GLOBALS
#include <includes.h>


#define ADC1_DR_Address    ((u32)0x4001244C)
extern unsigned int CCR2_Val_0;

/*
*********************************************************************************************************
*                                      LOCAL CONSTANTS
*********************************************************************************************************
*/
/*
*********************************************************************************************************
*                                      LOCAL DATA TYPES
*********************************************************************************************************
*/

/*
*********************************************************************************************************
*                                      LOCAL TABLES
*********************************************************************************************************
*/
/*
*********************************************************************************************************
*                                  LOCAL GLOBAL VARIABLES
*********************************************************************************************************
*/
static volatile ErrorStatus HSEStartUpStatus = SUCCESS;
/*
*********************************************************************************************************
*                                  LOCAL FUNCTION PROTOTYPES
*********************************************************************************************************
*/
static void SysTick_Config(void);
void GPIO_Config(void);
void SPI_Config(void);
void TIM7_Configuration(void);
void Timer_3(void);

/*
```

```
*********************************************************************************************************
*                                LOCAL CONFIGURATION ERRORS
*********************************************************************************************************
*/
/*
*********************************************************************************************************
**                                  Global Functions
*********************************************************************************************************
*/
/*
*********************************************************************************************************
*                                  BSP INITIALIZATION
* Description : This function should be called by your application code before you make use of any of the
*               functions found in this module.
* Arguments   : none
*********************************************************************************************************
*/

void  BSP_Init(void)
{
 /* SYSCLK, HCLK, PCLK2 and PCLK1 configuration ----------------------------*/
 /* RCC system reset(for debug purpose) */
 RCC_DeInit();

 /* Enable HSE */
 RCC_HSEConfig(RCC_HSE_ON);

 /* Wait till HSE is ready */
 HSEStartUpStatus = RCC_WaitForHSEStartUp();

 if(HSEStartUpStatus == SUCCESS)
  {
   /* Enable Prefetch Buffer */
   FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);

   /* Flash 2 wait state */
   FLASH_SetLatency(FLASH_Latency_2);
      /* HCLK = SYSCLK */
   RCC_HCLKConfig(RCC_SYSCLK_Div1);
    /* PCLK2 = HCLK */
   RCC_PCLK2Config(RCC_HCLK_Div1);
    /* PCLK1 = HCLK/2 */
   RCC_PCLK1Config(RCC_HCLK_Div2);

                    /* ADCCLK = PCLK2/4 */
                    RCC_ADCCLKConfig(RCC_PCLK2_Div6);

   /* PLLCLK = 8MHz * 9 = 72 MHz */
   RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);
    /* Enable PLL */
   RCC_PLLCmd(ENABLE);
    /* Wait till PLL is ready */
   while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET)
    {
    }
    /* Select PLL as system clock source */
```

```
    RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);

    /* Wait till PLL is used as system clock source */
    while(RCC_GetSYSCLKSource() != 0x08)
     {
     }
   }

   /* Enable GPIOA, GPIOB, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG and AFIO clocks */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |RCC_APB2Periph_GPIOC
       | RCC_APB2Periph_AFIO, ENABLE);

 // STM3210E_LCD_Init();
   /* Clear the LCD */
 // LCD_Clear(White);
   Lcd_Configuration();
  GPIO_Config();
           TIM7_Configuration();
           Timer_3();
  SPI_Config();
  SysTick_Config();                              /* Initialize the uC/OS-II tick interrupt               */
}


/*
*********************************************************************************************************
*                                     DISABLE ALL INTERRUPTS
*
* Description : This function disables all interrupts from the interrupt controller.
* Arguments   : None.
* Returns     : None.
*********************************************************************************************************
*/
void  BSP_IntDisAll (void)
{
  // CPU_IntDis();
}

/*
*********************************************************************************************************
**                                      uC/OS-II Timer Functions
*********************************************************************************************************
*/
/*
*********************************************************************************************************
*********
*                                     TICKER INITIALIZATION
*
* Description : This function is called to initialize uC/OS-II's tick source (typically a timer generating interrupts every 1
to 100 mS).
* Arguments   : none
* Note(s)     : 1) The timer is setup for output compare mode BUT 'MUST' also 'freerun' so that the timer
*               count goes from 0x00000000 to 0xFFFFFFFF to ALSO be able to read the free running count.
*               The reason this is needed is because we use the free-running count in uC/OS-View.
*********************************************************************************************************
*/
```

```
void  SysTick_Config(void)
{
   RCC_ClocksTypeDef  rcc_clocks;
   INT32U        cnts;

   RCC_GetClocksFreq(&rcc_clocks);

   cnts = (INT32U)rcc_clocks.HCLK_Frequency/OS_TICKS_PER_SEC;

   SysTick_SetReload(cnts);
   SysTick_CLKSourceConfig(SysTick_CLKSource_HCLK);
   SysTick_CounterCmd(SysTick_Counter_Enable);
   SysTick_ITConfig(ENABLE);
}

void GPIO_Config(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;

                GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_8 | GPIO_Pin_9;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;                //ÍÆÍìÊä³ö
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_Init(GPIOA, &GPIO_InitStructure);

                GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_1;
        GPIO_Init(GPIOB, &GPIO_InitStructure);

                GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_12 | GPIO_Pin_13;
                        //LED1
                GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
                GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
                GPIO_Init(GPIOA, &GPIO_InitStructure);

                /* ÅäÖÃÊý¾ÝIO Á¬½Óµ½GPIOC0 **********************/
                GPIO_InitStructure.GPIO_Pin = GPIO_Pin_14;
                GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
                GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
                GPIO_Init(GPIOC, &GPIO_InitStructure);

                /* ÅäÖÃ¿ØÖÆIO Á¬½Óµ½ c1**********************/
                GPIO_InitStructure.GPIO_Pin = GPIO_Pin_15 ;
                GPIO_InitStructure.GPIO_Mode =          GPIO_Mode_IPU;   //      GPIO_Mode_IPU
                GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;  /
                GPIO_Init(GPIOC, &GPIO_InitStructure);

                GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
                GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
                GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
                GPIO_Init(GPIOB, &GPIO_InitStructure);
                GPIO_PinRemapConfig(GPIO_PartialRemap_TIM3 , ENABLE);
}

void TIM7_Configuration(void)
{
        TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
```

```
        /* TIM7 clock enable */
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
        TIM_DeInit(TIM2);
        TIM_InternalClockConfig(TIM2);
        /* Time base configuration */
        TIM_TimeBaseStructure.TIM_Period = 0XFFFF;
        TIM_TimeBaseStructure.TIM_Prescaler =72-1;
        TIM_TimeBaseStructure.TIM_ClockDivision = 0;
        TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
        TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
        TIM_ARRPreloadConfig(TIM2, ENABLE);

        /* TIM1 enable counter */
//      TIM_Cmd(TIM1, ENABLE);
}

void Timer_3(void){

        TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
        TIM_OCInitTypeDef  TIM_OCInitStructure;
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
  /* Time base configuration */
  TIM_TimeBaseStructure.TIM_Period = 12000;       //10000;
  TIM_TimeBaseStructure.TIM_Prescaler = 2;          //1439;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseStructure.TIM_RepetitionCounter = 0x0;
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);

  /* Master Mode selection */
  TIM_SelectOutputTrigger(TIM3, TIM_TRGOSource_Update);

  /* TIM enable counter */
  TIM_Cmd(TIM3, ENABLE);

        /* Master Configuration in PWM1 Mode */
  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;
  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStructure.TIM_Pulse = 0;
  TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_Low;
  TIM_OC2Init(TIM3, &TIM_OCInitStructure);

}
void SPI_Config(void)
{
  GPIO_InitTypeDef  GPIO_InitStructure;
  SPI_InitTypeDef   SPI_InitStructure;

  //SPI1 Periph clock enable
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_SPI1,ENABLE);
   //SPI2 Periph clock enable
 // RCC_APB1PeriphClockCmd( RCC_APB1Periph_SPI2, ENABLE ) ;

 //Configure SPI1 pins: SCK, MISO and MOSI
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
```

```
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA,&GPIO_InitStructure);

  //Configure SPI1 pins: SCK, MISO and MOSI
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_Init(GPIOA,&GPIO_InitStructure);

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
  GPIO_Init(GPIOA,&GPIO_InitStructure);

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
  GPIO_Init(GPIOB,&GPIO_InitStructure);

  // SPI1 Config
  SPI_InitStructure.SPI_Direction = SPI_Direction_2Lines_FullDuplex;
  SPI_InitStructure.SPI_Mode = SPI_Mode_Master;
  SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b;
  SPI_InitStructure.SPI_CPOL = SPI_CPOL_Low;
  SPI_InitStructure.SPI_CPHA = SPI_CPHA_1Edge;
  SPI_InitStructure.SPI_NSS = SPI_NSS_Soft;   //SPI_NSS_Hard
  SPI_InitStructure.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_64;
  SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB;
  SPI_InitStructure.SPI_CRCPolynomial = 7;
  SPI_Init(SPI1,&SPI_InitStructure);

  // SPI1 enable
  SPI_Cmd(SPI1,ENABLE);
}

unsigned char SPI_WriteByte(unsigned char data)
{
 unsigned char Data = 0;

   //Wait until the transmit buffer is empty
  while(SPI_I2S_GetFlagStatus(SPI1,SPI_I2S_FLAG_TXE)==RESET);
  // Send the byte
  SPI_I2S_SendData(SPI1,data);

   //Wait until a data is received
  while(SPI_I2S_GetFlagStatus(SPI1,SPI_I2S_FLAG_RXNE)==RESET);
  // Get the received data
  Data = SPI_I2S_ReceiveData(SPI1);

  // Return the shifted data
  return Data;
}
void SpiDelay(unsigned int DelayCnt)
{
 unsigned int i;
 for(i=0;i<DelayCnt;i++);
```

```
}

u16 TPReadX(void)
{
  u16 x=0;
  TP_CS();
  SpiDelay(10);
  SPI_WriteByte(0x90);
  SpiDelay(10);
  x=SPI_WriteByte(0x00);
  x<<=8;
  x+=SPI_WriteByte(0x00);
  SpiDelay(10);
  TP_DCS();
  x = x>>3;
  return (x);
}

u16 TPReadY(void)
{
 u16 y=0;
  TP_CS();
  SpiDelay(10);
  SPI_WriteByte(0xD0);
  SpiDelay(10);
  y=SPI_WriteByte(0x00);
  y<<=8;
  y+=SPI_WriteByte(0x00);
  SpiDelay(10);
  TP_DCS();
  y = y>>3;
  return (y);
}
```

PROGRAMA: APP.C

```
/*
*********************************************************************************************************
*                                          EXAMPLE CODE
*                  (c) Copyright 2003-2006; Micrium, Inc.; Weston, FL
*            All rights reserved.  Protected by international copyright laws.
*            Knowledge of the source code may NOT be used to develop a similar product.
*            Please help us continue to provide the Embedded community with the finest
*            software available.  Your honesty is greatly appreciated.
*********************************************************************************************************
*/

/*
*********************************************************************************************************
*                                          EXAMPLE CODE
*                                    ST Microelectronics STM32
*                                          with the
*                                  STM3210B-EVAL Evaluation Board
* Filename      : app.c
* Version       : V1.00
```

```
* Programmer(s) : Brian Nagel
*********************************************************************************************************
*/

/*
*********************************************************************************************************
*                                            INCLUDE FILES
*********************************************************************************************************
*/

#include <includes.h>
#include "math.h"

/* ----------------- APPLICATION GLOBALS ------------------ */
static OS_STK AppTaskStartStk[APP_TASK_START_STK_SIZE];
static OS_STK AppTaskUserIFStk[APP_TASK_USER_IF_STK_SIZE];
static OS_STK AppTaskKbdStk[APP_TASK_KBD_STK_SIZE];
static OS_STK AppTaskAdcStk[APP_TASK_ADC_STK_SIZE];
static OS_STK AppTaskTimStk[APP_TASK_TIM_STK_SIZE];

//static  OS_EVENT      *AppUserIFMbox;

int ExTick;
int Flag_EN1 = 0, Flag_EN2 = 0, flag_Enganche = 0, Flag_Emerg = 0;
u32 time1=0;
float s=0.0, tanque_med = 0.0, tanque_med2 = 0.0;
float uT,iT0,eT0;
u32 uT_int;

unsigned int CCR2_Val_1 = 0;
extern unsigned int Set_point;

/*
*********************************************************************************************************
*                                      LOCAL FUNCTION PROTOTYPES
*********************************************************************************************************
*/

static void AppTaskCreate(void);
static void AppTaskStart(void *p_arg);

static void AppTaskUserIF(void *p_arg);
static void AppTaskKbd(void *p_arg);
static void AppTaskAdc(void *p_arg);
static void AppTaskTim(void *p_arg);

void TX_Moudle(void);
void Delay(vu32 nCount);

/*
*********************************************************************************************************
*                                              main()
* Description : This is the standard entry point for C code.  It is assumed that your code will call
*               main() once you have performed all necessary initialization.
* Arguments   : none
* Returns     : none
```

```
*********************************************************************************************************
*/

void MainTask(void);

int  main (void)
{
   INT8U  err;
            /* Set the Vector Table base location at 0x08000000 */
  NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x0);

  //  BSP_IntDisAll();                /* Disable all interrupts until we are ready to accept them */

   OSInit();                      /* Initialize "uC/OS-II, The Real-Time Kernel"            */
OSTaskCreateExt(AppTaskStart,(void *)0,(OS_STK *)&AppTaskStartStk[APP_TASK_START_STK_SIZE-
1],APP_TASK_START_PRIO,APP_TASK_START_PRIO,(OS_STK
*)&AppTaskStartStk[0],APP_TASK_START_STK_SIZE,(void
*)0,OS_TASK_OPT_STK_CHK|OS_TASK_OPT_STK_CLR);

#if (OS_TASK_NAME_SIZE > 13)
   OSTaskNameSet(APP_TASK_START_PRIO, "Start Task", &err);
#endif

   OSStart();                                     /* Start multitasking (i.e. give control to uC/OS-II)       */
}


/*
*********************************************************************************************************
*                                      STARTUP TASK
* Description : This is an example of a startup task.  As mentioned in the book's text, you MUST
*               initialize the ticker only once multitasking has started.
* Arguments   : p_arg   is the argument passed to 'AppTaskStart()' by 'OSTaskCreate()'.
* Returns     : none
* Notes       : 1) The first line of code is used to prevent a compiler warning because 'p_arg' is not
*                  used.  The compiler should not generate any code for this statement.
*********************************************************************************************************
*/

static  void  AppTaskStart (void *p_arg)
{
  (void)p_arg;

   BSP_Init();                           /* Initialize BSP functions                               */
   #if (OS_TASK_STAT_EN > 0)
    OSStatInit();                                  /* Determine CPU capacity                                 */
   #endif

//   AppUserIFMbox = OSMboxCreate((void *)0);              /* Create MBOX for communication between Kbd and
UserIF     */
   AppTaskCreate();                        /* Create application tasks

   while(DEF_TRUE)
            {
             /* Task body, always written as an infinite loop. */
              OSTaskSuspend(OS_PRIO_SELF);
```

```
            //OSTimeDlyHMSM(0,0,0,200);
           // GPIO_SetBits(GPIOC,GPIO_Pin_6);
            //OSTimeDlyHMSM(0,0,0,200);
           // GPIO_ResetBits(GPIOC,GPIO_Pin_6);
             /* ExTick=TPReadY();
              GUI_DispDecAt(ExTick,20,40,4);
             */

    }
}


/*
*********************************************************************************************************
*                                CREATE APPLICATION TASKS
* Description:  This function creates the application tasks.
* Arguments  :  none
* Returns    :  none
*********************************************************************************************************
*/

static  void  AppTaskCreate(void)
{
 INT8U  err;
 OSTaskCreateExt(AppTaskUserIF,(void *)0,(OS_STK *)&AppTaskUserIFStk[APP_TASK_USER_IF_STK_SIZE-
1],APP_TASK_USER_IF_PRIO,APP_TASK_USER_IF_PRIO,(OS_STK *)&AppTaskUserIFStk[0],
                APP_TASK_USER_IF_STK_SIZE,
                (void *)0,
                OS_TASK_OPT_STK_CHK|OS_TASK_OPT_STK_CLR);


#if (OS_TASK_NAME_SIZE > 8)
 OSTaskNameSet(APP_TASK_USER_IF_PRIO, "User I/F", &err);
#endif

 OSTaskCreateExt(AppTaskKbd,(void *)0,(OS_STK *)&AppTaskKbdStk[APP_TASK_KBD_STK_SIZE-
1],APP_TASK_KBD_PRIO,APP_TASK_KBD_PRIO,(OS_STK *)&AppTaskKbdStk[0],
                APP_TASK_KBD_STK_SIZE,
                (void *)0,
                OS_TASK_OPT_STK_CHK|OS_TASK_OPT_STK_CLR);

#if (OS_TASK_NAME_SIZE > 8)
 OSTaskNameSet(APP_TASK_KBD_PRIO, "Keyboard", &err);
#endif

 OSTaskCreateExt(AppTaskAdc,(void *)0,(OS_STK *)&AppTaskAdcStk[APP_TASK_ADC_STK_SIZE-
1],APP_TASK_ADC_PRIO,APP_TASK_ADC_PRIO,(OS_STK *)&AppTaskAdcStk[0],
                APP_TASK_ADC_STK_SIZE,
                (void *)0,
                OS_TASK_OPT_STK_CHK|OS_TASK_OPT_STK_CLR);

#if (OS_TASK_NAME_SIZE > 8)
 OSTaskNameSet(APP_TASK_KBD_PRIO, "Adc", &err);
#endif
```

```
 OSTaskCreateExt(AppTaskTim,(void *)0,(OS_STK *)&AppTaskTimStk[APP_TASK_TIM_STK_SIZE-
1],APP_TASK_TIM_PRIO,APP_TASK_TIM_PRIO,(OS_STK *)&AppTaskTimStk[0],
                APP_TASK_TIM_STK_SIZE,
                (void *)0,
                OS_TASK_OPT_STK_CHK|OS_TASK_OPT_STK_CLR);

#if (OS_TASK_NAME_SIZE > 8)
 OSTaskNameSet(APP_TASK_TIM_PRIO, "Tim", &err);
#endif
}


/*
*********************************************************************************************************
*                                    USER INTERFACE TASK
*
* Description : This task updates the LCD screen based on messages passed to it by AppTaskKbd().
* Arguments   : p_arg   is the argument passed to 'AppStartUserIF()' by 'OSTaskCreate()'.
* Returns     : none
*********************************************************************************************************
*/

static  void  AppTaskUserIF (void *p_arg)
{

 (void)p_arg;
  // GUI_Init();
            GPIO_SetBits(GPIOA,GPIO_Pin_2);
            GPIO_ResetBits(GPIOA,GPIO_Pin_8);
            GPIO_SetBits(GPIOB,GPIO_Pin_1);
  while(DEF_TRUE)
    {
              MainTask();
//            GUIDEMO_Touch();
    }
}


/*
*********************************************************************************************************
*                                    KEYBOARD RESPONSE TASK
* Description : This task monitors the state of the push buttons and passes messages to AppTaskUserIF()
* Arguments   : p_arg   is the argument passed to 'AppStartKbd()' by 'OSTaskCreate()'.
* Returns     : none
*********************************************************************************************************
*/

static  void  AppTaskKbd (void *p_arg)
{
   u8 tick=0;
  (void)p_arg;
   while(DEF_TRUE)
     {
                        tick++;
       OSTimeDlyHMSM(0,0,0,10);
              GUI_TOUCH_Exec();
```

```
                if(tick&0x10)
                {
                        GPIO_ResetBits(GPIOA,GPIO_Pin_3);
                }
                else
                {
                        GPIO_SetBits(GPIOA,GPIO_Pin_3);
                }
            ;
    }
}

/*
*********************************************************************************************************
*                              ADC TASK
* Description : This task monitors the state of the push buttons and passes messages to AppTaskUserIF()
* Arguments   : p_arg   is the argument passed to 'AppStartKbd()' by 'OSTaskCreate()'.
* Returns     : none
*********************************************************************************************************
*/

static  void  AppTaskAdc (void *p_arg)
{
        TIM_OCInitTypeDef  TIM_OCInitStructure;
                float a,b,c;                //constantes del PID
                float rT,eT,iT,dT,yT;     //variables de ecuaciones
                float max,min;              //límites máximo y mínimo de control.
                min=0.0;                  //inicialización variables
                max=55.0;
                a=0.1243;                   //constantes del PID
                b=0.0062;
                c=0.1215;
 (void)p_arg;

                GPIO_ResetBits(GPIOC, GPIO_Pin_14);//c0
                GPIO_ResetBits(GPIOC, GPIO_Pin_15);//c1
  while(DEF_TRUE)
  {
                        if (!GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_12))
                        {
                                GUI_Delay(20);
                                GPIO_ResetBits(GPIOA,GPIO_Pin_8);
                                GPIO_SetBits(GPIOB,GPIO_Pin_1);
                                Flag_Emerg = 0;
                                flag_Enganche = 0;
                                Flag_EN2 = 1;
                        }
                        else
                        {
                                Flag_Emerg = 1;
                        }
                        if(Flag_Emerg == 1)
                        {
                                if (!GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_1))
                                {
                                        GUI_Delay(20);
```

```
                GPIO_SetBits(GPIOA,GPIO_Pin_8);
                GPIO_ResetBits(GPIOB,GPIO_Pin_1);
                flag_Enganche = 1;
                Flag_EN1 = 1;
        }
        else
        {
                Flag_EN1 = 0;
        }

        if (!GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_0))
        {
                GUI_Delay(20);
                GPIO_ResetBits(GPIOA,GPIO_Pin_8);
                GPIO_SetBits(GPIOB,GPIO_Pin_1);
                flag_Enganche = 0;
                Flag_EN2 = 1;
        }
        else
        {
                Flag_EN2 = 0;
        }

//*************************************************************************************
//*********************************************  PID  *********************************
//*************************************************************************************
        if(flag_Enganche ==1)
        {
                rT=55-Set_point;
                yT=tanque_med2;

                eT=rT-yT;                       //Cálculo error
                iT=b*eT+iT0;                    //Cálculo del término integral
                dT=c*(eT-eT0);                  //Cálculo del término derivativo
                uT=iT+a*eT+dT;                  //Cálculo de la salida PID
                if (uT>max) {       //Salida PID si es mayor que el MAX
                        uT=max;}
                else {
                        if (uT<min){    //Salida PID si es menor que el MIN
                                uT=min;
                        }
                                }

                uT_int = (uT*10000)/55;
                CCR2_Val_1 =uT_int;

                if (uT>min && uT<max) {
                iT0=iT;                 //Guardar variables
                eT0=eT;
                }

                //*********** CONTROL VALVULA ON-OFF ****************
                if(yT <= rT)
                {
                        GPIO_ResetBits(GPIOA,GPIO_Pin_9);
                }
```

```
                                        else
                                        {
                                                GPIO_SetBits(GPIOA,GPIO_Pin_9);
                                        }


        //*****************************************************************************************
        //*********************************************  PWM  *********************************
        //*****************************************************************************************

                                        /* Master Configuration in PWM1 Mode */
                                        TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;
                                        TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                                        TIM_OCInitStructure.TIM_Pulse = CCR2_Val_1;
                                        TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_Low;
                                        TIM_OC2Init(TIM3, &TIM_OCInitStructure);
                                        OSTimeDlyHMSM(0,0,0,20);
                                }
                                else
                                {
                                        TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;
                                        TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                                        TIM_OCInitStructure.TIM_Pulse = 0;
                                        TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_Low;
                                        TIM_OC2Init(TIM3, &TIM_OCInitStructure);
                                        OSTimeDlyHMSM(0,0,0,20);
                                }
                         }
                        else
                        {
                                        TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;
                                        TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
                                        TIM_OCInitStructure.TIM_Pulse = 0;
                                        TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_Low;
                                        TIM_OC2Init(TIM3, &TIM_OCInitStructure);
                                        OSTimeDlyHMSM(0,0,0,20);
                        }
         }
}

void Delay(vu32 nCount)
{
        for(; nCount != 0; nCount--);
}
void TX_Moudle(void)
{
                GPIO_SetBits(GPIOC, GPIO_Pin_14);//D0            TRIGÀ-¸ß
                Delay(100);
                GPIO_ResetBits(GPIOC, GPIO_Pin_14);
}

/*
*********************************************************************************************************
*                       TIM TASK
*
* Description : This task monitors the state of the push buttons and passes messages to AppTaskUserIF()
```

```
* Arguments   : p_arg   is the argument passed to 'AppStartKbd()' by 'OSTaskCreate()'.
* Returns     : none
*********************************************************************************************************
*/

static  void  AppTaskTim (void *p_arg)
{
  (void)p_arg;

          GPIO_ResetBits(GPIOC, GPIO_Pin_14);//c0
          GPIO_ResetBits(GPIOC, GPIO_Pin_15);//c1
  while(DEF_TRUE)
    {
                              if(Flag_Emerg == 1)
                              {
                                        if(flag_Enganche ==1)
                                        {
                                                  TX_Moudle();
                                                  while(GPIO_ReadInputDataBit(GPIOC,GPIO_Pin_15)==0);

                                                  TIM_Cmd(TIM2, ENABLE);

                                                  while(GPIO_ReadInputDataBit(GPIOC,GPIO_Pin_15)==1);
                                                  TIM_Cmd(TIM2, DISABLE);

                                                  time1=TIM_GetCounter(TIM2);
                                                  TIM_SetCounter(TIM2,0x00);
                                                  tanque_med = 66-s;
                                                  if(tanque_med>=0)
                                                  {
                                                            tanque_med2 = tanque_med;
                                                  }
                                                  time1=0;
                                                  OSTimeDlyHMSM(0,0,0,200);
                                        }
                              }
    }
}
```