

ASCENSOR ATV71 CONTROLADO POR CODESYS

Elevator ATV71 controlled by codesys

JOSÉ ALEJANDRO FRANCO CALDERÓN*, MANUEL FELIPE RODRÍGUEZ PÉREZ**

Recibido: 14 de mayo de 2015. Aceptado: 14 de junio de 2015

RESUMEN

El presente artículo describe el desarrollo e implementación del control de un motor AC conectado a un encoder y a un variador de velocidad Altivar (ATV71) simulando un ascensor de carga, todo mediante una interfaz gráfica de usuario (HMI) desarrollada en Codesys con simulación de botones internos y externos.

Palabras clave: motor AC, encoder, ATV71, HMI, codesys.

ABSTRACT

This article describes the development and implementation of the control of an AC motor connected to an encoder and an Altivar (ATV71) simulating a freight elevator, all through a graphical user interface (HMI) developed simulation buttons Codesys internal and external.

Key words: motor AC, encoder, ATV71, HMI, Codesys.

I. INTRODUCCIÓN

Los objetivos del sistema están orientados a crear una interfaz gráfica de usuario que pueda simular el control y funcionamiento de un ascensor, controlando un motor AC conectado a un variador Altivar (ATV71) y a un encoder implementando técnicas de control de movimiento.

dispositivo, y comprende todos los puntos de contacto entre el usuario y el equipo.

Normalmente suelen ser fáciles de entender y fáciles de accionar, aunque en el ámbito de la informática es preferible referirse a que suelen ser «amigables e intuitivos».

1) *Funciones principales*

Las funciones principales son las siguientes:

- Puesta en marcha y apagado.
- Control de las funciones manipulables del equipo.

II. MARCO TEÓRICO

A. Interfaz gráfica de usuario [1]

Es el medio con el que el usuario puede comunicarse con una máquina, equipo, computadora o

* Ingeniero Electrónico de la Escuela Colombiana de Ingeniería “Julio Garavito”, especialista en diseño de aplicaciones para televisión digital interactiva y en administración de tecnologías de la información para la comunicación virtual de la Universidad Manuela Beltrán, Estudiante de maestría en ingeniería electrónica en la Escuela Colombiana de Ingeniería Julio Garavito. Docente investigador de la facultad de ingeniería adscrito al Grupo de Investigación y Desarrollo en Ingeniería de Sistemas - GIDIS de la Corporación Universitaria Republicana. Correo electrónico: alejing@gmail.com

** Ingeniero Electrónico, especialista en redes de telecomunicaciones graduado con la distinción de Tesis Laureada de la universidad Santo Tomas. Estudiante de maestría en ingeniería electrónica en la Escuela Colombiana de Ingeniería Julio Garavito. Actualmente funcionario del Instituto de Hidrología Meteorología y Medio Ambiente-IDEAM Ingeniero del Grupo de Automatización realizando, diseño, mantenimiento e instalación de estaciones hidrometeorológicas automáticas en todo el territorio nacional. Correo electrónico: manuelfelipe.rodriguez@hotmail.com

- Manipulación de archivos y directorios.
- Herramientas de desarrollo de aplicaciones.
- Comunicación con otros sistemas.
- Información de estado.
- Configuración de la propia interfaz y entorno.
- Intercambio de datos entre aplicaciones.
- Control de acceso.
- Sistema de ayuda interactivo.

2) Tipos

En las interfaces de usuario se pueden distinguir básicamente tres tipos:

- **Una interfaz de hardware**, a nivel de los dispositivos utilizados para ingresar, procesar y entregar los datos: teclado, ratón y pantalla visualizadora.
- **Una interfaz de software**, destinada a entregar información acerca de los procesos y herramientas de control, a través de lo que el usuario observa habitualmente en la pantalla.
- **Una interfaz de software-hardware**, que establece un puente entre la máquina y las personas, permite a la máquina entender la instrucción y al hombre entender el código binario traducido a información legible.

B. Altivar (ATV71) [2]

Es un variador de velocidad para aplicaciones tanto de torque constante como de alta potencia o complejas con una potencia que oscila entre los 0.37 kW hasta más allá de los 500 kW, algunas características son las siguientes:

- 200... 240 V monofásico
- 200... 240 V/380... 690 V trifásico, 50/60 Hz
- Regulación de velocidad mediante control vectorial de flujo, con o sin sensor
- Rango de velocidad: de 1 a 1.000 en modo de bucle cerrado con retroalimentación de codificador, de 1 a 100 en modo de bucle abierto
- Terminal gráfico: texto sencillo, botón de navegación, teclas de función configurables, menú «Simply Start», etc.
- Protección del motor y del variador
- «Power Removal», funciones de seguridad ATEX
- Más de 150 funciones disponibles: Regulador de PID, control de frenos adecuado para viajar, izamiento y movimientos de rotación, etc.
- Filtro EMC clase A integrado
- Modbus y CANopen integrados
- Tarjetas de extensión de E/S, tarjetas de interfaz para el codificador
- Tarjetas de comunicación: Fipio, Ethernet, Modbus Plus, Profibus DP, DeviceNet, Uni-Telway, INTERBUS
- Tarjeta programable Controller Inside

C. Codesys [3]

Es un entorno de desarrollo para la programación de controladores conforme con el estándar industrial internacional IEC 61131-3. El término CODESYS es un acrónimo y significa Sistema de Desarrollo de Controladores.

CODESYS es desarrollado y comercializado por la empresa de software alemán 3S-Smart de Soluciones de Software situado en la ciudad bávara de Kempten. La Primera Versión (1,0) fue creada en 1994.

CODESYS puede descargarse desde el sitio web de la compañía.

1) Lenguajes de programación

Los seis lenguajes de programación para aplicaciones vienen definidos en el IEC 61131-3 y es-

tán disponibles en el entorno de desarrollo de Codesys.

a) Lenguajes de texto

IL (lista de instrucciones): Es un lenguaje de programación parecido al lenguaje ensamblador.

ST (texto estructurado): Es similar a la programación en PASCAL o C.

b) Lenguajes gráficos

LD (Diagrama Ladder): Permite al programador combinar los contactos de relé y las bobinas. Es el lenguaje de Programación de PLC por excelencia.

FBD (diagrama de bloques de función): Permite al usuario programar rápidamente, tanto expresiones como en lógica booleana.

SFC (Bloques de función secuenciales): Es conveniente para los procesos de programación secuencial. Dispone también de un editor gráfico que no está definido en la norma IEC.

CFC (Continuous Function Chart): Es una especie de editor de FBD libre. Es un editor orientado a FBD donde las conexiones entre los entradas, salidas y los operadores se fijan automáticamente. Todas las cajas se pueden colocar libremente, lo que permite programar ciclos de retroalimentación provisional sin variables.

2) *Uso industrial*

Más de 250 fabricantes de dispositivos de diferentes sectores industriales ofrecen sus dispositivos de automatización inteligente programable con la interfaz de programación CODESYS. En consecuencia, miles de usuarios finales en todo el mundo emplean CODESYS para su trabajo diario en todo tipo de tareas de automatización. Hoy en día, CODESYS es la herramienta de desarrollo basada en IEC 61131-3 más extendida en Europa.

Una red mundial de asociados del sistema de CODESYS ofrece tanto una amplia variedad de servicios para los usuarios CODESYS como el apoyo a los usuarios finales, soporte, consultoría, for-

mación, programación de aplicaciones o la integración de sistemas.

D. SoMove [4]

Es un software de configuración para ordenadores muy fácil de usar, que permite configurar los dispositivos de control del motor de Schneider Electric.

El software SoMove incorpora diversas funciones para la configuración de los dispositivos, como:

- Preparación de la configuración
- Configuración
- Mantenimiento

Para facilitar la configuración y el mantenimiento, SoMove

- Puede utilizar un enlace directo por cable USB/RJ45.
- Puede utilizar un enlace inalámbrico Bluetooth®.
- Es compatible con la herramienta de configuración Multi-Loader y con SoMove Mobile para teléfonos móviles.

Estas herramientas pueden ahorrar un tiempo considerable a la hora de cargar, duplicar o editar configuraciones en un dispositivo.

Un modo realmente autónomo permite realizar lo siguiente:

- Preparación de archivos de configuración
- Gestión y almacenamiento de archivos (guardado en disco duro o CD ROM, copia, cambio de nombre y envío por correo electrónico, etc.)
- Impresión de la lista de parámetros
- Preparación de archivos para las herramientas MultiLoader y SoMove Mobile

El modo con conexión se utiliza para las siguientes funciones:

- Configuración, ajuste, control y supervisión
- Transferencia de archivos de configuración entre SoMove V1.0 y el variador o el arrancador progresivo

E. Encoder [5]

Pueden monitorear electrónicamente la posición de un eje giratorio.

Los encoders absolutos son dispositivos electro-mecánicos es decir, que son elementos de retroalimentación útiles en sistemas de control de bucle cerrado. Proporcionan control de posición en aplicaciones de empaquetado, robótica, recogida y colocación, tornillo guía/ de bolas, posicionamiento de mesa rotativa e inserción de componentes.

Los encoders ópticos incrementales ofrecen bajo costo, tamaño físico más pequeño, alta frecuencia y alta resolución. Nuestros accesorios le ayudan a instalar fácilmente nuestros encoders y a usarlos con eficiencia.

F. Control de movimiento [6]

El control de movimiento de motores hace referencia al comportamiento físico que realiza un motor cuando está en marcha basado en parámetros como velocidad, aceleración y posicionamiento.

Generalmente el control de movimiento es todo un sistema basado en componentes físicos y lógicos como por ejemplo:

- **Software de aplicación:** herramienta lógica en la que se puede indicar posiciones deseadas y perfiles de control de movimiento.
- **Controlador de movimiento:** Es el cerebro del sistema. Toma los perfiles de las posiciones y movimientos deseados y crea las trayectorias que deberán seguir los motores. Entregando señales de voltaje a servomotores; o pulsos de paso y dirección, a motores de pasos.
- **Amplificador o drive:** Los amplificadores (también llamados drives) toman los comandos del controlador y generan la corriente necesaria para dirigir o girar el motor.
- **Motor:** Encargado de convertir la energía eléctrica en energía mecánica y producir el torque requerido para moverse a la posición deseada.
- **Elementos mecánicos:** Los motores están diseñados para proporcionar torque a algunos dispositivos mecánicos. Éstos incluyen deslizadores lineales, brazos robóticos y actuadores especiales.
- **Dispositivo de retroalimentación o sensor de posición:** El dispositivo de retroalimentación, generalmente un codificador de cuadratura (encoder), detecta la posición del motor y reporta el resultado al controlador, y de esa manera cierra el lazo con el controlador de movimiento.

1) Perfil de velocidad

A fin de alcanzar movimientos suaves de alta velocidad sin forzar el motor, un controlador de movimiento utiliza los valores de la posición especificada deseada, la velocidad máxima que se desea alcanzar y la aceleración para determinar cuánto tiempo empleará en los tres segmentos de movimiento principales (los cuales incluyen aceleración, velocidad constante y desaceleración).

En las aplicaciones típicas de control de motores, el movimiento comienza desde una posición de detenimiento o desde un movimiento previo y sigue una rampa de aceleración indicada hasta que la velocidad alcanza la velocidad deseada, el movimiento continúa a esta velocidad por un periodo de tiempo hasta que el controlador determina que es tiempo de comenzar el segmento de desaceleración, y desciende la velocidad del movimiento para detenerse exactamente en la posición deseada. Este comportamiento es conocido como perfil de velocidad trapezoidal el cual es mostrado en la figura 1.

Si el movimiento es lo suficientemente corto de manera que el punto de comienzo de la desaceleración ocurre antes de que se haya completado el aceleramiento, entonces el perfil toma una forma triangular en vez de trapezoidal y la velocidad alcanzada podría ser menor a la velocidad que se deseaba alcanzar. En la figura 2 se muestra un ejemplo de perfil triangular.

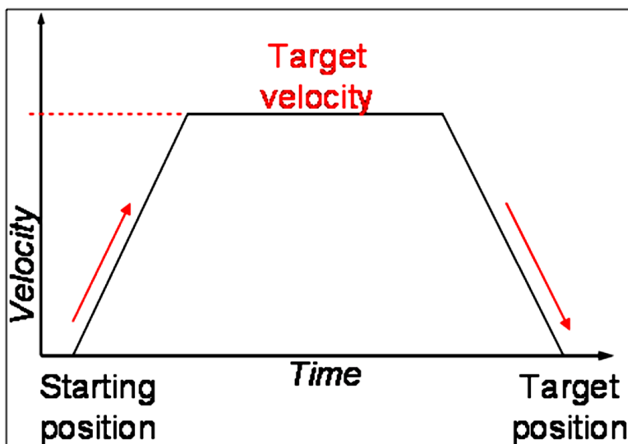


Figura 1. Perfil de velocidad trapezoidal.

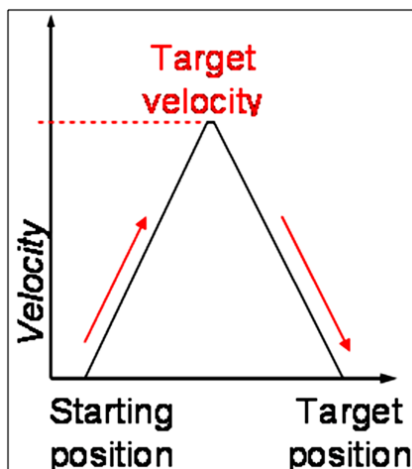


Figura 2. Perfil de velocidad triangular.

III. CÁLCULOS TEÓRICOS Y PRUEBAS

Para el diseño del ascensor se tuvo en cuenta como parámetro inicial que de acuerdo a las dimensiones físicas de la planta existente con la cual se trabajó, el ascensor debería ser de 3 pisos o niveles entre los cuales variaría el movimiento.

A. Cálculo de pulsos de encoder por piso

Para realizar el cálculo de los pulsos del encoder por piso se determinó la altura inicial del ascensor y con la ayuda del software programado en Codesys se midieron los pulsos del encoder que desde el piso inicial hasta la altura máxima del piso 3, este procedimiento se realizó 5 veces a fin de determinar aproximadamente la altura total

del ascensor en pulsos del encoder obteniendo en promedio 220000 Pulsos del encoder. En la siguiente Tabla se puede observar los datos prácticos obtenidos durante el cálculo de la altura del ascensor.

Tabla 1. Datos de la Altura total del Ascensor.

Movimiento	Posición
1	220148
2	220053
3	220183
4	219863
5	219627
Promedio	219974,8 » 220000

Por lo tanto dividiendo el total de pulsos en 3 pisos obtenemos que:

$$\begin{aligned} \text{Pulsos por piso} &= \frac{220000 \text{ pulsos}}{3 \text{ pisos}} = 73333,333 \\ &\approx 73333 \frac{\text{pulsos}}{\text{piso}} \end{aligned}$$

Con el valor obtenido anteriormente podemos definir cuantos pulsos del encoder corresponden a cada piso, valores que son mostrados en la siguiente tabla.

Tabla 2. Pulsos correspondientes a cada piso.

Piso	Pulsos por Piso
1	0
2	73333
3	146666

B. Cálculo de pulsos de encoder para parar en un piso

Una vez definido los pulsos que debe registrar el encoder por cada piso recorrido el siguiente paso fue determinar en cuantos pulsos del encoder se debe dar la orden de parada en subida para que se posicionen justamente en los pulsos determinados por piso. Se realizaron 5 pruebas de paradas en el segundo piso dando la orden de parada justamente en **73333 pulsos**, los resultados obtenidos se observan en la siguiente tabla.

Tabla 3. Datos de posición de parada a 73333 pulsos.

Movimiento	Posición Parada
1	78968
2	78978
3	78992
4	78290
5	78216
Promedio	78688,8 » 78689

Con el valor promedio obtenido de los datos de la tabla anterior se obtiene la diferencia que hay entre la posición final de parada y la posición en la que se genera la orden, por lo tanto:

$$\text{Diferencia} = 78689 - 73333 = 5356 \text{ pulsos}$$

Con esta diferencia y los pulsos correspondientes a cada piso se puede determinar en qué posición exacta se debe dar la orden de parada del motor para que pare exactamente en el piso correspondiente. Es importante tener en cuenta para el piso 2 que la orden de parada debe considerarse si el movimiento es en subida o si es en bajada. En la siguiente tabla se relacionen los valores de posición a los cuales se debe dar la orden de parada para cada piso.

Tabla 4. Orden de parada en cada piso.

Piso	Orden de Parada
1 (Bajada)	5356
2 (Subida)	67977
2 (Bajada)	78689
3 (Subida)	141310

C. Cálculo del torque máximo soportado

En vista que la planta del ascensor permite variar el peso de carga con un juego de pesas, uno de los requisitos de funcionamiento hace referencia la detección de sobrepeso a través del torque que genera el motor.

Por criterio de diseño se decide que la operación normal del ascensor sea con un peso de 40lb (2 pesas) y que la interfaz gráfica muestre una alerta de sobrepeso cuando el ascensor tenga un peso mayor o igual a 100lb (5 pesas) por lo cual se decide realizar 8 movimientos de prueba con los cuales

se obtienen los datos mostrados en la siguiente tabla:

Tabla 5. Torque generado a diferentes pesos en el motoR

Movimiento	40 lb	100 lb
1	29	42
2	28	40
3	28	40
4	27	41
5	27	40
6	27	41
7	28	39
8	27	40
Promedio	28	40

Basados en los resultados obtenidos determinamos como constante de torque máximo 40Nm.

IV. DESARROLLO

El desarrollo del proyecto supuso la integración de todo un sistema desde el software de la interfaz gráfica de usuario hasta la puesta en marcha del actuador (motor AC) conectado a un ascensor de carga.

A. Resumen de la aplicación

Objetivo general de la aplicación: Definir una posición de inicio que corresponde al piso 1, la aplicación mostrará el tablero interior y exterior de un ascensor la cual permitirá mover el ascensor a la posición deseada.

Variables a controlar: Posición.

Parámetros variar por el usuario final: Piso dentro del ascensor y solicitud del mismo desde cada piso.

Tipo de motor: A.C.

Dispositivo a utilizar para realizar el control de movimiento: Variador de velocidad con Controller Inside.

Protocolo de comunicación a utilizar con el computador: Modbus

B. Descripción del sistema

A continuación un gráfico que describe la integración implementada. Ver figura 3.

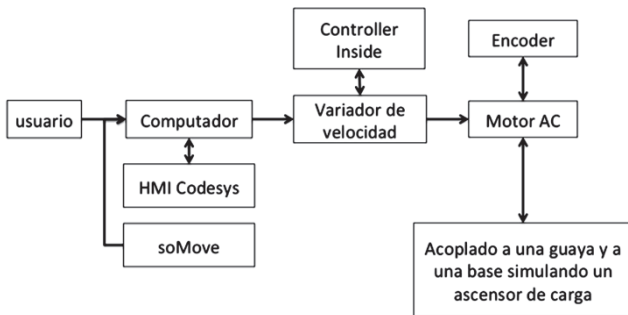


Figura 3. Gráfico de la integración implementada.

El usuario interactúa mediante una interfaz gráfica (HMI) desarrollada en Codesys y un computador con el ascensor. Tanto el computador como el software Codesys están conectados al controller inside del variador de velocidad mediante un protocolo de comunicación RS485 a RS232.

El software SoMove se utilizó para determinar el mapa de memoria que se podía usar y poder interactuar con el mediante otra aplicación de lectura y escritura de datos posiblemente desarrollada en LabVIEW. Este software se sincronizó modo modbus con el controller inside que posee el variador y se pudo así determinar la pila de memoria que se programaría para lograr la integración.

El controller inside acoplado al variador Altivar ATV71 permitía a su vez controlar el adecuado funcionamiento de del motor AC.

El motor AC a su vez tenía acoplado un encoder a su eje con el cual se nos permitiría leer los pulsos que emitía por revolución y así determinar la posición.

Con el cálculo tanto de la posición como del torque lograríamos así controlar piso a piso el ascensor y simular una posible sobrecarga del peso en el mismo.

1) Descripción detallada de la solución

a) Parámetros de configuración

Para el adecuado funcionamiento del variador que controla el motor a continuación se eviden-

cian los parámetros configurados en el Altivar ATV71.

Ajustes del motor (Figura 4 y 5).



Figura 4. Configuración de parámetros del motor a controlar [Potencia, voltaje, corriente y frecuencia].



Figura 5. Configuración de parámetros del motor a controlar [Velocidad nominal del motor, frecuencia máxima y autoajuste]

Control del motor por rampa trapezoidal



Figura 6. Configuración de la rampa que controlará el motor desde el variador [Incremento, tiempo de la rampa de aceleración y desaceleración, velocidad mínima y máxima].

Comunicación Codesys y ATV71

- Carta de programación 1

Tabla 6. Parámetros de comunicación Codesys & ATV71 RS485 a RS232.

Parámetro	Valor
Velocidad	19200b
Paridad	No
Bits	8
Bit parada	1

Comunicación soMove y Controller Inside ATV71

- Carta de programación 4

Tabla 7. Parámetros de comunicación soMove & Controller Inside ATV71 RS485 a RS232 (Modbus).

Parámetro	Valor
Velocidad	38400b
Paridad	No
Bits	8
Bit parada	1

b) Funciones

Las funciones que se implementaron para poder controlar el motor AC son las siguientes:

Drive Actual Position Get (TRUE)

Permite obtener la posición actual por ciclo asíncrono, para este caso la lectura del encoder se realizó cada 100 ms (Figura 7).

```

Timer1(IN:=(NOT Timer1.Q),PT:=T#100ms);
IF Timer1.Q THEN
%MW71:=ActTorque;
%MW72:=REAL_TO_WORD(SQRT(ActVel*ActVel));
%MW73:=DRIVE_PI1;
%MW74:=DRIVE_PI2;
%MW75:=DRIVE_PI3;

cont2:=cont2+1;
END_IF

```

Figura 7. Captura de la posición como tarea asíncrona cada 100 ms.

Drive Actual Velocity Get (TRUE)

Permite obtener la velocidad actual que lleva el motor en RPM para posteriormente ubicarlo en algún indicador.

Drive Actual Torque Get (TRUE)

Devuelve el torque que una carga esta ejerciendo sobre el motor para posteriormente ubicarlo en algún indicador.

Drive Run Forward ()

Pone en marcha a la velocidad dispuesta en frecuencia al motor hacia delante, para el caso de la aplicación para arriba.

Drive Run Reverse ()

Pone en marcha a la velocidad dispuesta en frecuencia al motor hacia atrás, para el caso de la aplicación para abajo.

Drive Stop Ramp ()

Para el motor en rampa sin permitir un frenado busco con alto impacto.

c) Estrategia de software

Para el adecuado funcionamiento de la aplicación la estrategia de software se baso en dos variables importantes el torque y la posición.

Con la posición en pulsos de encoder se pudo determinar con cuantos de ellos el ascensor se movería de piso en piso.

Lo primero que se realizó fue una definición de variables que se usarían a lo largo del programa se puede ver en la figura 8.

Con las variables definidas se procedió a definir una velocidad constante para el ascensor así

```

0001 PROGRAM CommandATV71
0002 VAR
0003     ButtonStopPushed: BOOL;
0004     VizuVelocity: INT;
0005     actualposition: DWORD:=0;
0006     direcion: BOOL;
0007     actual2ente: INT;
0008     resta: INT;
0009     res: INT;
0010     reset: BOOL;
0011     actualpositionres: BOOL;
0012     primero: BOOL;
0013     stop1: BOOL;
0014     segundo: BOOL;
0015     stop2: BOOL;
0016     tercero: BOOL;
0017     stop3: BOOL;
0018     indicador1: BOOL;
0019     indicador2: BOOL;
0020     indicador3: BOOL;
0021     sentido:BOOL;
0022     PisoActual:INT;
0023     torquemax:INT;
0024     torquetotal:INT;
0025     primeroex: BOOL;
0026     segundoex: BOOL;
0027     terceroex: BOOL;
0028     externo1: BOOL;
0029     externo2: BOOL;
0030     externo3: BOOL;
0031     sobrepeso: BOOL;
0032     resetpeso: BOOL;
0033 END_VAR
    
```

Figura 8. Variables utilizadas en el programa.

como la variable que sería el valor del torque (peso) máximo que soportaría el mismo, esto se puede ver en la figura 9.

```

VizuVelocity:=250; (*Constante de Velocidad*)
torquemax:=40; (*Torque maximo para sobrepeso*)
    
```

Figura 9. Máxima velocidad y peso que el ascensor soporta.

Se implemento un botón de parada de emergencia con el cual se podría detener la aplicación en cualquier momento si existía algún problema, esto se puede evidenciar en la figura 10.

```

IF ButtonStopPushed OR %MW52>0 THEN (*Boton de parada de emergencia en rueda libre*)
DriveStopFreeWheel(); (*Activa variable para condicion de parada en el piso 2*)
END_IF
    
```

Figura 10. Programación del botón parada de emergencia.

Mediante el cálculo de la posición actual menos la anterior se podía determinar la posición en cada instante del ascensor, esto se programo como lo evidencia la figura 11.

```

(*Calculo de la posicion actual*)
res:=actual2ente;
actualposition:=DriveActualPositionGet(TRUE);
actual2ente:=DWORD_TO_INT(actualposition);
direcion:=DriveStatusGet.bDirection;
resta:=actual2ente-res;

pos:=pos+resta; (*Variable de visualizacion de la posicion en pulsos del encoder*)
%MW54:=DINT_TO_WORD(pos); (*Asignacion de la variable posicion al mapa de memoria*)
    
```

Figura 11. Cálculo de la posición actual del ascensor.

Posteriormente se programó el botón reset que inicializaba en cero el valor de pulsos del encoder cuando el ascensor se encontraba en el primer piso, a demás permitía inicializar la lógica de los indicadores de piso, el número del piso en el que se encontraba y el torque máximo que por defecto era cero (ascensor sin carga), esto se puede ver en la figura 12.

```

(*Boton de Posicion Inicial, reinicio de variables a estado inicial *)
IF reset OR %MW53>0 THEN
pos:=0; (*coloca posicion inicial en 0 pulsos del encoder*)
sentido:=TRUE; (*Cambia indicador de sentido a subida*)
indicador1:=TRUE; (*Activa indicador de piso 1*)
indicador2:=FALSE; (*Apaga indicador de piso 2*)
indicador3:=FALSE; (*Apaga indicador de piso 3*)
PisoActual:=1; (*Cambia indicador de piso actal a 2*)
sobrepeso:=FALSE; (*Apaga el indicador de sobrepeso*)
torquetotal:=0; (*Coloca torque total en 0*)
END_IF
    
```

Figura 12. Programación del botón reset (inicio del sistema).

A continuación se le envía la orden al variador para que trabaje a una determinada velocidad previamente definida como una constante global, así como la definición y lectura de las variables velocidad actual y torque actual, ello se evidencia en la figura 13.

```
(*Asignacion de la constante de velocidad al movimineto en el variador*)
DriveTargetVelocitySel(VizuVelocity);
(*Asignacion de variable de de Velocidad Actual*)
ActVel:=DriveActualVelocityGet(TRUE);10;
(*Asignacion de variable de de Torque Actual*)
ActTorque:=DriveActualTorqueGet(TRUE);
```

Figura 13. Variables torque actual, velocidad actual y asignación de velocidad constante.

La estrategia entonces ahora pasaba por determinar 2 cosas: según el piso presionado por el usuario se determinaría primero en que rango de posición estaba para saber hacia donde debía moverse el ascensor (arriba o abajo) y segundo calcular el torque acumulado para saber si sobrepasaba nuestro limite de simulación máximo que era 40 Nm, evidencia de la programación de ello se puede ver en la figura 14.

```
IF ActTorque>torquetotal THEN (*Visualizacion del torque total del movimiento*)
torquetotal:=ActTorque;
%MW70:=torquetotal; (*Asignacion de la variable de torque total al mapa de memoria*)
END_IF

IF torquetotal>=torquemax THEN (*Condicion de torque para indicar sobrepeso y parar en rueda libre*)
DriveStopFreeWheel(); (*Orden de parada de movimiento en rueda libre*)
sobrepeso:=TRUE; (*Activacion del Indicador de Sobrepeso*)
END_IF

IF resetpeso OR %MW51>0 THEN (*Boton para reiniciar en caso de sobrepeso*)
torquetotal:=0;
sobrepeso:=FALSE; (*Apaga el indicador de sobrepeso*)
END_IF
```

Figura 14. Acumula y actualiza el indicador con el troque máximo que lleva el ascensor.

La figura 15 a continuación muestra como si se ha superado el troque dispuesto como máximo para la simulación la interfaz cambia según lo dispuesto en la programación mostrada en la figura 14.

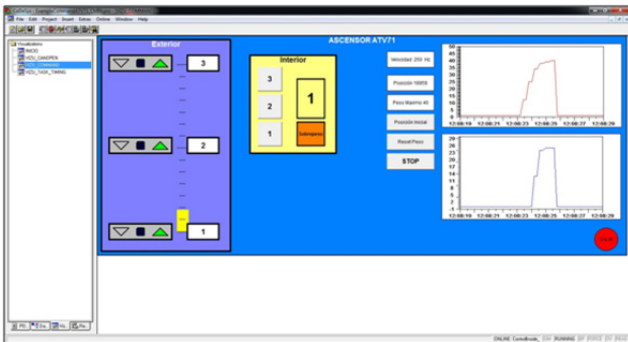


Figura 15. HMI con el torque máximo superado e indicador sobrepeso activo en naranja. Ascensor detenido.

Si el torque máximo no era superado entonces debíamos monitorear en que sentido estaba el ascensor, si estaba subiendo el programa tenia una variable booleana llamada sentido, donde si esta estaba en TRUE el ascensor iba para arriba de lo contrario iba para abajo.

Identificando luego del sentido y la posición donde se encontraba el ascensor se procedía entonces a ir donde el usuario interno o externo lo deseara.

Para determinar el largo de pulsos que se tenia para la aplicación se tomo desde el rango de 0 a 14666 pulsos, y estos se dividieron entre 2 para determinar a que equivalía cada piso. Es decir que desde el pulso 0 hasta el 73333 el ascensor se encontraba entre el piso 1 y 2 parando en el piso 2 (73333 pulsos), y si el ascensor estaba entre los pulsos 73333 hasta 14666 el ascensor se encontraba entre los pisos 2 y 3 con parada en el piso 3 (14666 pulsos). Con estos rangos, el sentido (subida o bajada), la validación del botón de piso presionado (interno o externo), la validación del torque máximo y el calculo evidenciado en los cálculos teóricos para la dar la orden de parada al ascensor sin que sobrepasara el limite de piso definido se programo el controller inside.

A continuación la verificación programada cuando el ascensor se encontraba en el primer piso, y lo que debía hacer según el usuario y el estado del ascensor.

```
(* Verificacion de la posicion del ascensor dentro del piso 1*)
IF pos>=0 AND pos<73333 AND sentido=TRUE THEN

(* Acciones ejecutadas al presionar el Boton interior piso 2 ó solicitud externa del piso 2*)
IF (segundo OR segundox OR %MW55>0 OR %MW56>0) AND NOT sobrepeso THEN
indicador1:=FALSE; (*Apaga indicador de piso 1*)
stop2:=TRUE; (*Activa variable para condicion de parada en el piso 2*)
DriveRunForward(); (*Inicia movimiento del motor hacia arriba*)
END_IF

(*Condicion para detener el movimiento en el piso 2*)
IF pos>=67977 AND stop2=TRUE AND sentido=TRUE THEN
stop2:=FALSE; (*Apaga variable para condicion de parada en el piso 2*)
DriveStopRamp(); (*Orden de parada de movimiento de por rampa*)
indicador2:=TRUE; (*Activa variable indicador de piso 2*)
PisoActual:=2; (*Cambia indicador de piso actual a 2*)
END_IF

(* Acciones ejecutadas al presionar el Boton interior piso 3 ó solicitud externa del piso 3*)
IF (tercero OR tercerox OR %MW57>0 OR %MW58>0) AND NOT sobrepeso THEN
indicador1:=FALSE; (*Apaga indicador de piso 1*)
stop3:=TRUE; (*Activa variable para condicion de parada en el piso 3*)
DriveRunForward(); (*Inicia movimiento del motor hacia arriba*)
END_IF

(*Condicion para detener el movimiento en el piso 3*)
IF pos>141310 AND stop3=TRUE AND sentido=TRUE THEN
stop3:=FALSE; (*Apaga variable para condicion de parada en el piso 3*)
DriveStopRamp(); (*Orden de parada de movimiento de por rampa*)
indicador3:=TRUE; (*Activa variable indicador de piso 3*)
PisoActual:=3; (*Cambia indicador de piso actual a 3*)
END_IF
END_IF
```

Figura 16. HMI Ascensor en el primer piso y la lógica de funcionamiento si va al segundo o tercer piso.

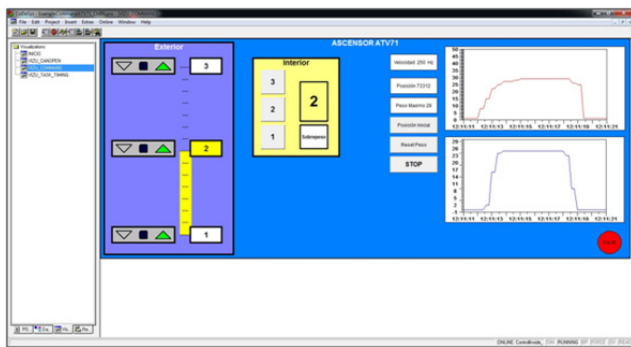


Figura 17. HMI Ascensor controlado desde el interior del piso 1 al piso 2, indicador de subida activo más graficas de velocidad y torque.

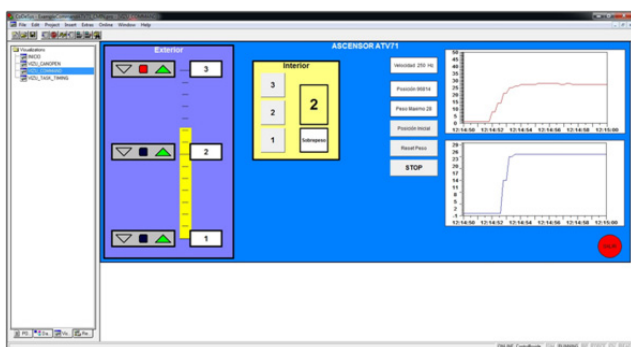


Figura 18. Ascensor controlado desde el exterior del piso 2 al piso 3, indicador de subida activo más graficas de velocidad y torque.

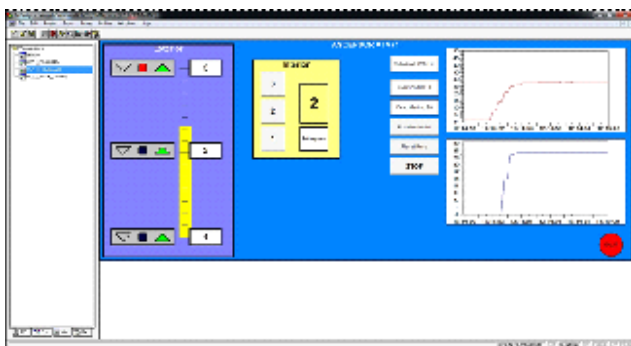


Figura 19. HMI Ascensor controlado desde el exterior del piso 1 al piso 3, indicador de subida activo más graficas de velocidad y torque.

A continuación la verificación programada cuando el ascensor se encontraba en el segundo piso, y lo que debía hacer según el usuario y el estado del ascensor.

```
(* Verificación de la posición del ascensor dentro del piso 2 y sentido en subida*)
IF pos>67977 AND pos<146666 AND sentido=TRUE THEN
    PisoActual:=2; (*Cambia indicador de piso actual a 2*)

    (* Acciones ejecutadas al presionar el Boton interior piso 3 ó solicitud externa del piso 3*)
    IF (tercero OR terceroex OR %MW57>0 OR %MW58>0) AND NOT sobrepeso THEN
        stop3:=TRUE; (*Activa variable para condición de parada en el piso 3*)
        DriveRunForward(); (*Inicia movimiento del motor hacia arriba*)
        indicador2:=FALSE; (*Apaga indicador de piso 2*)
    END_IF

    (*Condición para detener el movimiento en el piso 3*)
    IF pos>=141310 AND stop3=TRUE AND sentido=TRUE THEN
        stop3:=FALSE; (*Apaga variable para condición de parada en el piso 3*)
        DriveStopRamp(); (*Orden de parada de movimiento de por rampa*)
        indicador3:=TRUE; (*Activa variable indicador de piso 3*)
        PisoActual:=3; (*Cambia indicador de piso actual a 3*)
    END_IF

    (* Acciones ejecutadas al presionar el Boton interior piso 1 ó solicitud externa del piso 1*)
    IF (primero OR primeroex OR %MW59>0 OR %MW60>0) AND NOT sobrepeso THEN
        stop1:=TRUE; (*Activa variable para condición de parada en el piso 1*)
        DriveRunReverse(); (*Inicia movimiento del motor hacia abajo*)
        indicador2:=FALSE; (*Apaga indicador de piso 2*)
        sentido:=FALSE; (*Cambia indicador de sentido a bajada*)
    END_IF
END_IF
```

Figura 20. Ascensor en el segundo piso y la lógica de funcionamiento si va al tercero o primero

```
(* Verificación de la posición del ascensor dentro del piso 3*)
IF pos>141310 THEN
    PisoActual:=3; (*Cambia indicador de piso actual a 3*)
    sentido:=FALSE; (*Cambia indicador de sentido a bajada*)

    (* Acciones ejecutadas al presionar el Boton interior piso 2 ó solicitud externa del piso 2*)
    IF (segundo OR segundoex OR %MW55>0 OR %MW56>0) AND NOT sobrepeso THEN
        stop2:=TRUE; (*Activa variable para condición de parada en el piso 2*)
        DriveRunReverse(); (*Inicia movimiento del motor hacia abajo*)
        indicador3:=FALSE; (*Apaga indicador de piso 3*)
    END_IF

    (* Acciones ejecutadas al presionar el Boton interior piso 1 ó solicitud externa del piso 1*)
    IF (primero OR primeroex OR %MW59>0 OR %MW60>0) AND NOT sobrepeso THEN
        stop1:=TRUE; (*Activa variable para condición de parada en el piso 1*)
        DriveRunReverse(); (*Inicia movimiento del motor hacia abajo*)
        indicador3:=FALSE; (*Apaga indicador de piso 3*)
    END_IF
END_IF
```

Figura 21. Ascensor en el tercer piso y la lógica de funcionamiento si va al segundo o primero

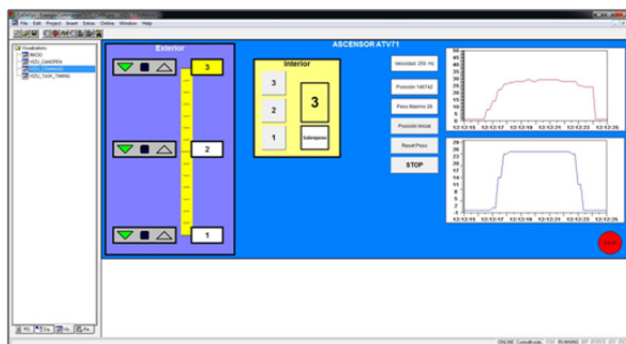


Figura 22. HMI Ascensor controlado desde el interior del piso 2 al piso 3, indicador de subida bajando debido a que esta llegando al piso 3 y no hay mas pisos, más graficas de velocidad y torque.

A continuación la verificación programada cuando el ascensor se encontraba tanto en el tercer piso como en el segundo piso (bajando), y lo que debía hacer según el usuario y el estado del ascensor.

```
(* Verificación de la posición del ascensor dentro del piso 3 y sentido en bajada*)
IF pos>67977 AND pos<146666 AND sentido=FALSE THEN

(*Condicion para detener el movimiento en el piso 2 y sentido en bajada*)
IF pos<=78689 AND stop2=TRUE AND sentido=FALSE THEN
  DriveStopRamp(); (*Orden de parada de movimiento de por rampa*)
  indicador2:=TRUE; (*Activa variable indicador de piso 2*)
  PisoActual:=2; (*Cambia indicador de piso actual a 2*)
  stop2:=FALSE; (*Apaga variable para condicion de parada en el piso 2*)
END_IF

(*Condicion para detener el movimiento en el piso 1 y sentido en bajada*)
IF pos<=5356 AND stop1=TRUE AND sentido=FALSE THEN
  stop1:=FALSE; (*Apaga variable para condicion de parada en el piso 1*)
  DriveStopRamp(); (*Orden de parada de movimiento de por rampa*)
  indicador1:=TRUE; (*Activa variable indicador de piso 1*)
  PisoActual:=1; (*Cambia indicador de piso actual a 1*)
  sentido:=TRUE; (*Cambia indicador de sentido a subida*)
END_IF
END_IF
```

Figura 23. Ascensor en el tercer piso y la lógica de funcionamiento si va al segundo o primero

```
(* Verificación de la posición del ascensor dentro del piso 2 y sentido en bajada*)
IF pos>=0 AND pos<73333 AND sentido=FALSE THEN

(*Condicion para detener el movimiento en el piso 1 y sentido en bajada*)
IF pos<=5356 AND stop1=TRUE AND sentido=FALSE THEN
  stop1:=FALSE; (*Apaga variable para condicion de parada en el piso 1*)
  DriveStopRamp(); (*Orden de parada de movimiento de por rampa*)
  indicador1:=TRUE; (*Activa variable indicador de piso 1*)
  PisoActual:=1; (*Cambia indicador de piso actual a 1*)
  sentido:=TRUE; (*Cambia indicador de sentido a subida*)
END_IF
END_IF
```

Figura 24. Ascensor en el segundo piso y la lógica de funcionamiento si va al primero

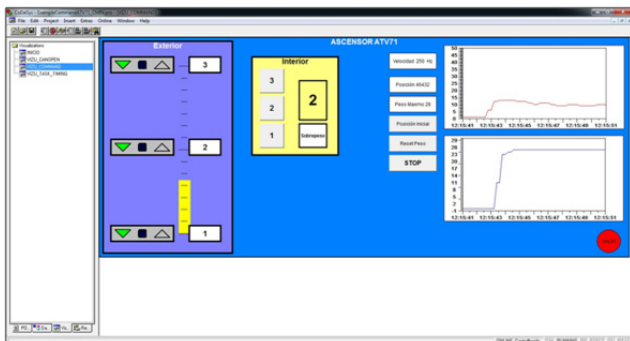


Figura 25. HMI Ascensor controlado desde el interior del piso 3 al piso 1, indicador de subida bajando, más graficas de velocidad y torque.

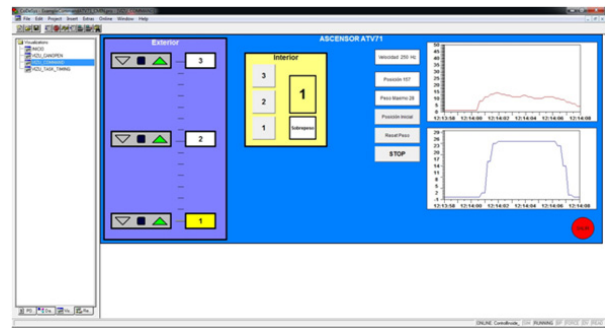


Figura 26. HMI Ascensor controlado desde el exterior del piso 2 al piso 1, indicador de subida [subiendo] ya que no hay más pisos para donde bajar, más graficas de velocidad y torque.

Por último se procedió a programar la lógica de los botones e indicadores externos del ascensor lo cual funcionaba para identificar en el programa cuando se pedía el ascensor desde el interior y cuando no y así se lograr un funcionamiento adecuado, evidencia de ello se puede ver en la figura 26.

```
(* CAMBIO DE ESTADO INDICADOR EXTERNO*)

(* Acciones ejecutadas al presionar el boton de solicitud externa del piso 1*)
IF (primeros OR %MV60=0) AND sobrepeso=FALSE THEN
  externo1:=TRUE; (*Activa variable indicador externo de piso 1*)
END_IF
IF PisoActual=1 THEN (* Acciones ejecutadas al llegar al piso 1*)
  externo1:=FALSE; (*Apaga variable indicador externo de piso 1*)
END_IF
%MV64=BOOL_TO_WORD(externo1); (*Asignacion de la variable de indicador de solicitud de piso externo al mapa de memoria*)

(* Acciones ejecutadas al presionar el boton de solicitud externa del piso 2*)
IF (segundos OR %MV55=0) AND sobrepeso=FALSE THEN
  externo2:=TRUE; (*Activa variable indicador externo de piso 2*)
END_IF
IF PisoActual=2 THEN (* Acciones ejecutadas al llegar al piso 2*)
  externo2:=FALSE; (*Apaga variable indicador externo de piso 2*)
END_IF
%MV65=BOOL_TO_WORD(externo2); (*Asignacion de la variable de indicador de solicitud de piso externo al mapa de memoria*)

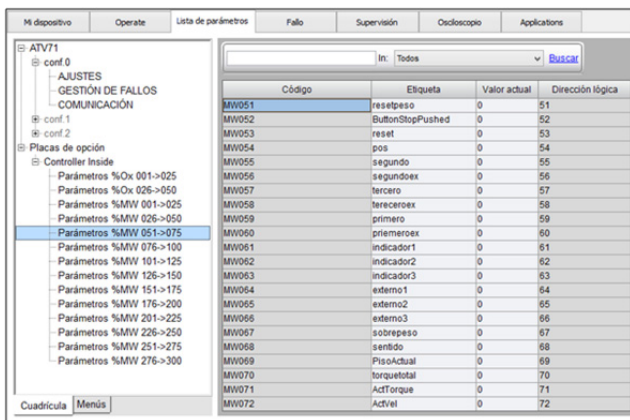
(* Acciones ejecutadas al presionar el boton de solicitud externa del piso 3*)
IF (terceros OR %MV58=0) AND sobrepeso=FALSE THEN
  externo3:=TRUE; (*Activa variable indicador externo de piso 3*)
END_IF
IF PisoActual=3 THEN (* Acciones ejecutadas al llegar al piso 3*)
  externo3:=FALSE; (*Apaga variable indicador externo de piso 3*)
END_IF
%MV66=BOOL_TO_WORD(externo3); (*Asignacion de la variable de indicador de solicitud de piso externo al mapa de memoria*)
```

Figura 27. Lógica de programación de los botones e indicadores del control externo del ascensor

d) Mapa de memoria

Con el fin de poder controlar el ascensor desde una interfaz externa por medio del protocolo de comunicaciones MODBUS, dentro del programa se asigna a cada una de las variables de control e indicadores una posición de memoria correspondiente a los parámetros del controller inside.

Con la ayuda del software SoMove podemos observar los parámetros del mapa de memoria los cuales tienen asignada una etiqueta nombrada igual a las variables utilizadas en el programa, en la siguiente figura se puede observar la lista de parámetros en SoMove.



Código	Etiqueta	Valor actual	Dirección lógica
MW051	resetpeso	0	51
MW052	ButtonStopPushed	0	52
MW053	reset	0	53
MW054	pos	0	54
MW055	segundo	0	55
MW056	segundex	0	56
MW057	tercero	0	57
MW058	tercerorex	0	58
MW059	primero	0	59
MW060	primerorex	0	60
MW061	indicador1	0	61
MW062	indicador2	0	62
MW063	indicador3	0	63
MW064	extemo1	0	64
MW065	extemo2	0	65
MW066	extemo3	0	66
MW067	sobrepeso	0	67
MW068	sentido	0	68
MW069	PisoActual	0	69
MW070	torqueTotal	0	70
MW071	ActTorque	0	71
MW072	ActVel	0	72

Figura 28. Lista de parámetros (Mapa de Memoria) del controller inside en SoMove

e) Video de la aplicación funcional

La aplicación funcionando se puede observar en el siguiente enlace al video publico en YouTube.

<https://www.youtube.com/watch?v=vO45IsL-Fjn4feature=youtu.be>

V. CONCLUSIONES

Se cumple con el objetivo general del proyecto ya que mediante técnicas aplicadas del control de movimiento se brinda una solución eficaz enfocada a movimientos verticales de tipo ascensor.

Codesys es un software de desarrollo con grandes capacidades ya que además de permitir la programación interna del controller inside también permitió desarrollar una interfaz gráfica la cual es amigable para cualquier persona, y para este caso simula el interior y exterior de un ascensor real, adicionalmente se pueden visualizar gráficas de comportamiento de la velocidad y torque del motor en cada movimiento.

Fue necesario implementar un botón de reinicio de sobrepeso ya que al no tener un sensor de peso real y de medición continua el sistema no puede detectar en tiempo real cuando se ha ganado o a perdido peso hasta que pase un tiempo y se supere el torque máximo establecido.

Para la medición de la posición del motor es indispensable contar con un encoder y un acople

adecuado, así como las tarjetas de lectura para el variador de velocidad y el controller inside.

Los cálculos realizados para el diseño y funcionamiento adecuado del ascensor fueron tomados experimentalmente realizando movimientos aleatorios y obteniendo promedios para mejorar la precisión en el posicionamiento de cada uno de los pisos.

Con la ayuda del variador de velocidad se pueden desarrollar diferentes aplicaciones donde se requiera tener velocidades altas con movimientos suaves sin necesidad de forzar el motor y producir averías en este.

En la gráfica de comportamiento de la señal de velocidad del motor, se puede observar claramente un perfil de velocidad tipo trapezoidal donde las rampas de aceleración y desaceleración coinciden con lo establecido dentro de los parámetros configurados en el variador.

A cada una de los botones e indicadores que aparecen en la interfaz gráfica se le asigna una posición en el mapa de memoria del controller inside, permitiendo que se pueda tener el control de movimiento por medio de una interfaz externa con comunicación vía MODBUS al modificar los valores de cada uno de los parámetros establecidos para el ascensor.

La versatilidad de tener un mapa de memoria organizado en el controller inside es que permite comunicar otro tipo de interfaces de usuario ya sea en lenguajes de programación de alto nivel o de consola, todo según la aplicación para la cual se desea diseñar dicha interfaz (HMI).

VI. REFERENCIAS

- [1] Interfaz de usuario. Wikipedia. Recuperado el 05.12.2015 de https://es.wikipedia.org/wiki/Interfaz_de_usuario
- [2] Schneider Electric. Altivar ATV 71. Recuperado el 05.12.2015 de <http://www.schneider-electric.com/products/co/ls/2900-motion-drives/2950-variadores-de-velocidad-multi-aplicacion/1155-altivar-71/>
- [3] Codesys. Wikipedia. Recuperado el 05.12.2015 de <https://es.wikipedia.org/wiki/CoDeSys>

- [4] Rockwell Automation. Encoders. Recuperado el 05.12.2015 de <http://ab.rockwellautomation.com/es/Motion-Control/Encoders>
- [5] Schneider Electric. soMove. Recuperado el 05.12.2015 de <http://www.schneider-electric.com/products/es/es/5100-software/5105-software-de-configuracion/2714-somove/>
- [6] National Instruments. Fundamentos de control de movimiento. Recuperado el 05.12.2015 de <http://www.ni.com>