UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DESARROLLO DE FIRMWARE Y SOFTWARE NECESARIO PARA LA ACTUALIZACIÓN DE LAS CÁMARAS TRIAXIALES DE LA COORDINACIÓN DE GEOTECNIA DEL INSTITUTO DE INGENIERÍA DE LA UNAM

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Milton Octavio García Martínez

DIRECTOR DE TESIS

Mtro. Miguel Ángel Mendoza García

Ciudad Universitaria, Cd. Mx., 2019

Agradecimientos

A ti especialmente Aline, por estar siempre a mi lado, por acompañarme, apoyarme, ver y vivir de cerca este largo y sinuoso camino, ayudarme a salir adelante en cualquier situación. Por todas y cada una de las clases que tomamos juntos, por los proyectos, series, exámenes, nuestros picnics y caminatas al terminar las clases que compartimos. En fin, por tantas cosas tan únicas y maravillosas que vivimos juntos durante toda nuestra carrera, las cuales sin duda fueron la motivación más importante para culminarla. ♡ ¡Te AMo Mucho! AA&&MO 4374r && P.S. ♡

A mis padres por todo el apoyo que me han brindado, por su paciencia para poder verme como un ingeniero, por los valores inculcados y la formación que se encargaron de darme para poder hacer posible esto.

A mi hermana por apoyar mis ocurrencias, por estar siempre dispuesta a ayudarme, por motivarme a ser un buen hermano. Por todas y cada una de las risas que me ha brindado incondicionalmente.

A toda mi familia (abuelos, abuelas, tíos, tías, primos, primas, sobrinos, sobrinas), por estar siempre al pendiente de mi trayectoria académica.

A la familia Lozano Moctezuma por ser testigo de esta formación y por todo el apoyo que me han brindado.

A mis amigos por haberme considerado para formar parte de algo tan grande y trabajar conjuntamente para seguir creciendo.

A la Universidad Nacional Autónoma de México, por todo, "por mi raza hablará el espíritu".

A la Facultad de Ingeniería, por sus amplias y valiosas enseñanzas.

A mi jurado conformado por:

- Ing. Eduardo Carranza Torres
- M.I. Miguel Angel Mendoza García
- Dr. Eduardo Espinoza Ávila
- Dr. Guillermo Gilberto Molero Castillo
- Dr. Ismael Everardo Barcenas Patiño

Gracias por su tiempo y conocimientos para hacer esto posible.

Gracias Miguel por el apoyo y la paciencia para desarrollar este proyecto.

Gracias Lalo por todas las asignaturas impartidas y por demostrarme que la computación es única e irremplazable.

Índice general

Índice de figuras

Índice de tablas

Índice de códigos

Capítulo I

Introducción

El mezclar diferentes ramas de ingeniería entre sí o con otras disciplinas puede resultar en algo sorprendente, tal es el caso de la ingeniería biomédica, en donde se unen ingeniería y medicina para dar como resultado aquella rama tan innovadora y de gran importancia para nuevos desarrollos tecnológicos.

En diversas ocasiones una mezcla puede parecer imposible o absurda dada la complejidad que podría conllevar desarrollar algo con base en esta unión de disciplinas. Sin embargo, basta con dar una oportunidad para poner a prueba alguna combinación.

La coordinación de electrónica y la de geotecnia, del Instituto de Ingeniería, desarrollaron esta oportunidad creando un proyecto encargado de mezclar la ingeniería en computación con la mecánica de suelos.

Lo que se requería para este proyecto era actualizar un equipo para pruebas de mecánica de suelos llamado cámara triaxial, el cual ya contaba con cierta electrónica, pero la puesta en marcha de este desarrollo lo controlaría prácticamente por completo mediante una tarjeta desarrollada por el coordinador de electrónica del IINGEN.

La cámara triaxial debía ser controlada mediante un firmware y un software, y fue lo que se elaboró: un firmware embebido en la tarjeta controladora capaz de manejar diversos estados que la cámara triaxial podía presentar; así como un software que se comunicara y controlara la tarjeta mediante un panel que reemplazaría los sistemas neumáticos, como manómetros de carátula y reguladores analógicos de presión.

1.1. Objetivo

Desarrollar firmware y software necesario para la actualización de las cámaras triaxiales ubicadas en el laboratorio de geotecnia del instituto de Ingeniería de la UNAM. Lo cual parte de la necesidad de este para actualizar algunos sistemas con los que cuenta desde hace aproximadamente 60 años, adaptándose así, al cambio tecnológico y aprovechando la posibilidad de actualizarlos con un menor costo que el de adquirir equipos modernos. Es decir, que se está haciendo un sistema a la medida, reduciendo costos y apoyando la tecnología elaborada en casa.

1.2. Hipótesis

Es posible elaborar firmware y software, para controlar una cámara triaxial desde una computadora mediante la utilización de una tarjeta electrónica con un micro-controlador embebido, así como otros componentes electrónicos y mecánicos.

1.3. Organización del documento

Esta tesis consta de cuatro temas principales en donde se abordan todos los pasos que lograron cumplir con el objetivo planteado. Los temas son los siguientes:

Antecedentes: aquí se abordan los conceptos de mayor importancia para el desarrollo de este proyecto y que se utilizan en múltiples ocasiones dada la naturaleza del mismo. Todo con la finalidad de contextualizar el desarrollo.

Firmware: dado que la tarjeta controladora proporcionada es la parte clave de este desarrollo, en este capítulo se explica el código elaborado, los requerimientos que el sistema debía cumplir, como se comporta el firmware, la forma en que controla la tarjeta y finalmente como interactúa con el software.

Interfaz de usuario (software): ya que este desarrollo no solo constaba de un firmware, sino de un software que interactuara con él, también se llevó a cabo la elaboración de una interfaz gráfica capaz de interactuar con el firmware y a su vez controlar la tarjeta.

Integración y pruebas con el sistema final: una vez que se conocieron los conceptos previos, se desarrolló el firmware y software, se prosiguió a la integración con el sistema controlador de la cámara triaxial, tanto en el aspecto físico como en el digital.

Capítulo II

Antecedentes

En el presente capítulo se abordarán los conceptos más importantes utilizados en el desarrollo del proyecto presentado, con la finalidad de poner en contexto las cosas que podrían considerarse como parte medular de lo realizado y que además fueron de utilidad para desarrollar el proyecto.

De igual forma, se llevará a cabo un pequeño recorrido a modo introductorio para conocer los conceptos que giran en torno a la tarjeta controladora con la que se trabajó y las herramientas extras utilizadas para lograr que llevara a cabo las tareas para las cuales fue concebida

2.1. Cámara triaxial

Dado que el objetivo de la tesis consiste en actualizar las cámaras triaxiales del laboratorio de mecánica de suelos del Instituto de Ingeniería de la Universidad Nacional Autónoma de México, es de suma importancia explicar a grandes rasgos ¿qué son?, ¿para qué sirven?, y la trascendencia que dichos equipos representan en la actualidad.

Según el sitio del laboratorio de mecánica de suelos [12], este laboratorio (véase figura 2.1) depende de la Coordinación de Geotecnia del Instituto de Ingeniería, inicio sus actividades en 1958 y se encuentra ubicado en el Edificio 4, Raúl Marsal Córdova"del II UNAM, Ciudad Universitaria.

Entre los diversos equipos con los que cuenta dicha coordinación, se encuentran los siguientes tipos de cámaras triaxiales:

Figura 2.1: Interior del área de mecánica de suelos

- Cámaras triaxiales dinámicas para probetas de 3.6 y 7 cm de diámetro.
- Cámaras triaxiales estáticas de desplazamiento controlado.

Para la actualización, únicamente se utilizará una cámara triaxial del tipo "estática de desplazamiento controlado" (figura 2.2), aunque el programa realizado será capaz de controlar hasta 4 equipos de características similares.

Y, ¿qué es una cámara triaxial? Según Zea [20, 2004, pp. 58-59], "es una cámara de compresión en forma cilíndrica que se llena con agua por medio de la cual se puede aplicar presión al interior de la cámara. Consta de una base metálica, una camisa transparente, una tapa metálica y tornillos de sujeción. Además de contar con otra base de menor diámetro donde se coloca la muestra cilíndrica que se va a ensayar."

2.1.1. Usos

¿Para qué sirve una cámara triaxial? En esencia, sirve para llevar a cabo un proceso experimental por medio del cual se somete una muestra cilíndrica de suelo a una

condición tal de esfuerzo que se asemeje a la recibida en su entorno natural, recibiendo distintos tipos de esfuerzos hasta que dicha muestra falle. Se conoce como triaxial porque la muestra está sometida a esfuerzos y deformaciones tridimensionales [21].

Dado que la resistencia del suelo, a diferencia de otros materiales, no es una propiedad invariable ya que oscila dentro de unos amplios limites dependiendo el caso. La cámara triaxial (figura 2.2) provee las herramientas para llevar a cabo un ensayo triaxial, el cual es un procedimiento para medir la resistencia al esfuerzo cortante de un suelo. La cámara permite controlar las tensiones principales, lo cual provoca tener conocimiento del comportamiento elemental del suelo.

Figura 2.2: Cámara triaxial del laboratorio de mecánica de suelos

2.2. Microcontrolador

El microcontrolador, es un circuito integrado, con capacidad de ser programado y de ejecutar las instrucciones que contenga en su memoria [19]. Un microcontrolador posee una arquitectura básica, la cual consta de los siguientes elementos:

- Unidad central de procesamiento.
- Memoria.
- Periféricos de entrada y salida.

Su arribo a la vida diaria marcó un hito importante para las diversas actividades que se realizaban. Ya que permitió reducir considerablemente la utilización de circuitos lógicos, dado que se trata de un elemento programable, con memoria propia, económico y de bajo consumo energético.

Actualmente los microcontroladores tienen una amplia gama de aplicaciones, desde electrodomésticos, radios hasta automóviles o la industria aeronáutica. Gracias al avance tecnológico, los microcontroladores actuales poseen tamaños cada vez más reducidos y con un incremento considerable en su potencia de procesamiento, además de presentar mejoras en sus prestaciones.

Y, ¿por qué trabajar con un microcontrolador y no con un microprocesador? En este desarrollo se trabajó con un microcontrolador dado que este ya se encontraba en la tarjeta controladora, sin embargo, las ventajas que esto presenta es que puede trabajar de cierta manera independiente, ya que como se explicó más arriba, el microcontrolador posee diversas características que lo hacen tener prácticamente todo lo necesario para funcionar, dado que se puede ver como una computadora en un solo circuito.

Una razón más por la cual la utilización del microcontrolador (figura 2.3) es una buena elección, es porque la cantidad de memoria y capacidad de procesamiento no es necesariamente alta, para esta aplicación se utilizará para acciones en específico, las cuales no requieren de tanto procesamiento como el de un microprocesador.

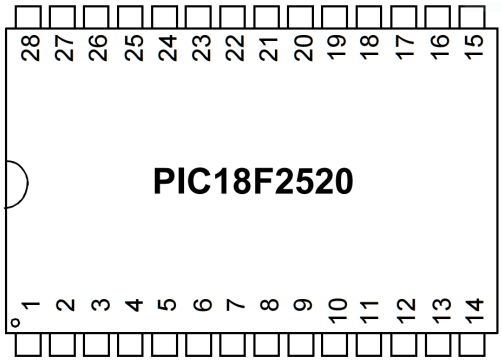

Figura 2.3: Esquema del microcontrolador utilizado, recuperado de [17]

2.3. RS485

Se trata de una especificación de la norma RS449 publicada en 1983 como estándar, especificando sus características eléctricas en un circuito de comunicación digital. Una de estas es que posee tres estados lógicos “1” o “0”, el tercer estado surge de una señal diferencial más una señal de alta impedancia, lo cual provoca que se pueda conectar más de un emisor en la red.

Este estándar define conexiones con un cable de par trenzado de cobre y con terminales RJ11, lo cual otorga mayor resistencia a la interferencia electromagnética y una mejora en velocidad de transmisión que la norma RS232.

RS-485 no define un protocolo; simplemente una interfaz eléctrica. Aunque muchas aplicaciones utilizan niveles de señal RS-485, la velocidad, el formato y el protocolo de la transmisión de datos no están especificados en RS-485. La interoperabilidad incluso de dispositivos similares de diferentes fabricantes no está garantizada por el solo cumplimiento de los niveles de señal.

Para este desarrollo se utiliza el protocolo RS485 de manera que pueda intercomunicar las diversas tarjetas controladoras que se pueden conectar en el mismo canal de datos, solo una de las tarjetas deberá conectarse a una computadora mediante un adaptador RS232 a RS485, provocando una disminución en costos de implementación y mejorando la comunicación entre dispositivos.

Cada tarjeta posee un chip RS485, el cual se encuentra conectado directamente al microcontrolador, esto logra una comunicación directa y disminuye la probabilidad de que la información llegue a perderse en el camino.

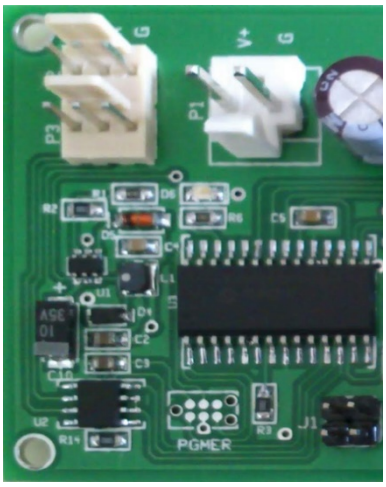

Figura 2.4: Detalle del microcontrolador conectado al convertidor RS48

En la figura 2.4 se muestra como están interconectados los dispositivos para interactuar, del microcontrolador al convertidor y del convertidor a las terminales de salida (esquina superior izquierda), de las cuales una se conectará a la computadora mediante un adaptador y la otra se interconectará con otra tarjeta controladora.

2.4. USB

La palabra USB hace referencia a la abreviatura en ingles de Universal Serial Bus. El cual es un estándar que establece especificaciones para cables, conectores y protocolos. Fue lanzado en el año de 1996 y actualmente se encuentra en su tercera generación.

El campo de aplicación es bastante amplio, en la actualidad existen cada vez más dispositivos con dicho puerto, desde un reproductor multimedia hasta ciertos modelos de automóviles.

Dado que la tarjeta controladora cuenta con un puerto RS485 y como se buscaba que fuera controlada desde una computadora con la que contaba el laboratorio de mecánica de suelos, se requirió un adaptador USB a RS485, el cual se encarga de realizar el cambio de una señal a otra y que ambos dispositivos se pudieran comunicar. El conversor USB a RS485 (figura 2.5) integra un conversor USB a serial y un chip serial a RS485, permitiendo comunicar una computadora y la tarjeta.

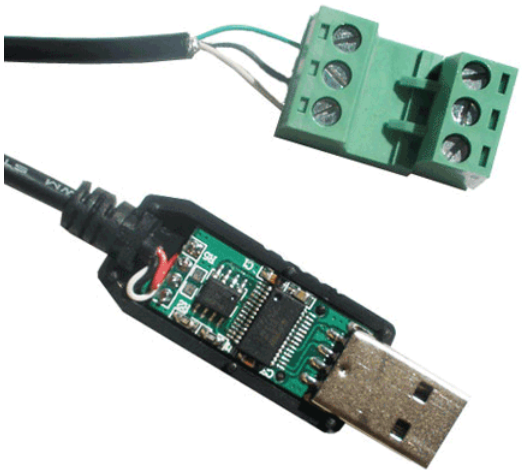

Figura 2.5: Cable convertidor de USB a RS485, obtenido de [14]

2.5. DAC

Debido a que la tarjeta controladora depende en gran medida de la utilización de un DAC, se consideró pertinente explicar a grandes rasgos que es un DAC y cómo funciona.

Por sus siglas en inglés (Digital to Analog Converter), o convertidor digital a analógico, es un dispositivo que transforma datos digitales en una señal analógica (figura 2.6). Un DAC puede reconstruir señales digitales en una señal analógica con determinada precisión, esto dependiendo del convertidor utilizado. Los datos digitales en esta aplicación se generan en el microcontrolador, listos para llevar a cabo la conversión a una señal analógica para interactuar con el mundo real, o su equivalente, una servo válvula encargada de proveer de una determinada presión relacionada a un valor digital generado por el microcontrolador.

Figura 2.6: Ilustración de una conversión de señal digital a señal analógica

2.6. Compilador

Por haberse desarrollado un firmware para la tarjeta controladora usando indirectamente un compilador, se considera importante el dar una breve introducción sobre dicha herramienta, con la cual el desarrollo de firmware se simplifica, dado que provee de un conjunto de instrucciones especializadas relacionadas en este caso para el microcontrolador.

Se puede definir al compilador como un programa que traduce código fuente escrito en algún lenguaje de programación de alto nivel a su equivalente en código de máquina para determinada arquitectura.

El compilador utilizado en este desarrollo se trata de uno llamado PHC de la empresa CCS Inc. Este compilador esta específicamente diseñado para satisfacer las necesidades del microcontrolador PIC. Y según la información en la página web del fabricante, "permite a los desarrolladores diseñar rápidamente software de aplicaciones en un lenguaje más legible y de alto nivel" [10].

Así mismo, permite implementar de forma eficiente y directa códigos escritos en lenguaje de programación C, ya que es posible efectuar operaciones de entrada y salida, utilización de matrices, y utilizar todos los tipos de datos nativos del lenguaje C.

2.7. Tarjeta controladora

Desarrollada en la coordinación de electrónica del Instituto de ingeniería, la tarjeta es capaz de llevar a cabo comunicaciones con múltiples dispositivos (figura 2.7), controlar periféricos y además almacenar datos de configuración en memoria no volátil en caso de alguna falla de alimentación o algún problema similar.

A pesar de sus reducidas dimensiones físicas de aproximadamente 11 [cm] x 5 [cm], en esta tarjeta se hacen presentes diversos periféricos y circuitos integrados, encargados de realizar las funciones necesarias para las que fue creada.

Cabe destacar que toda la programación se adaptó a la tarjeta, debido a que fue suministrada por la coordinación de electrónica del Instituto de Ingeniería de la UNAM.

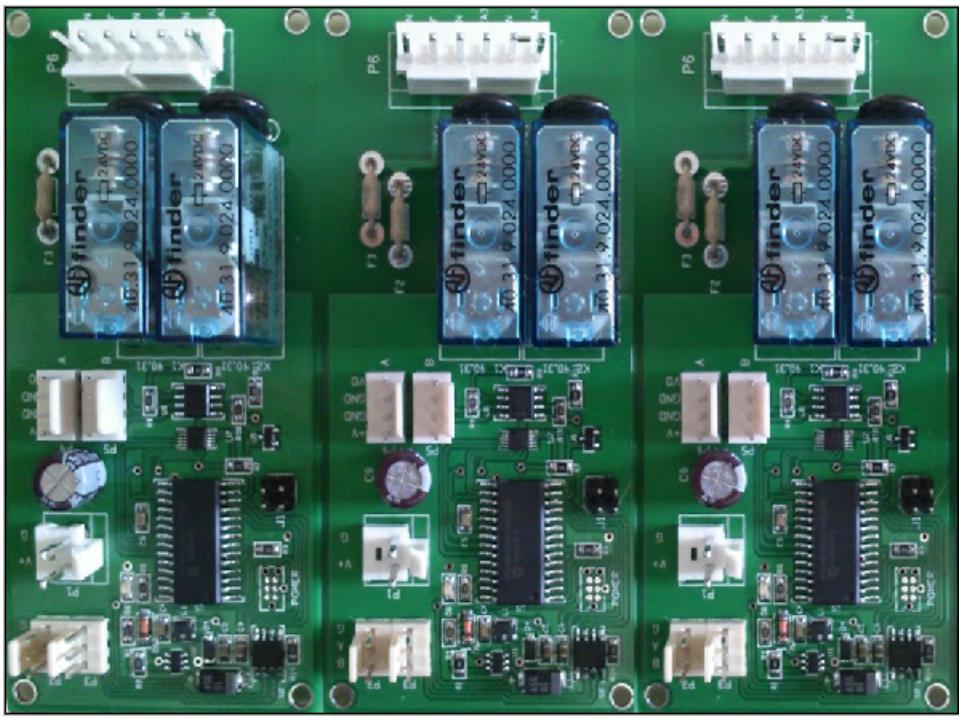

Figura 2.7: Conjunto de tarjetas controladoras listas para ser programadas

2.7.1. Programación de la tarjeta

Para poder utilizar la tarjeta controladora para los fines que fue creada, no bastaba contar con una electrónica capaz de realizar dicho propósito, era necesario el poder implementar un firmware encargado de hacer uso de todos sus periféricos de manera adecuada y con la finalidad de poder utilizarse en varias tarjetas elaboradas con el mismo propósito.

Gracias a la versatilidad de los microcontroladores, para las primeras pruebas realizadas no fue necesario contar con la tarjeta final a la que se cargaría el programa definitivo. Afortunadamente eso facilitó el poder conocer la plataforma, ya que la experiencia con la que se contaba para emprender el proyecto era elemental.

Si bien las primeras pruebas fueron lentas, conforme se fue conociendo la plataforma a utilizar y las herramientas con las que se trabajaría, el proceso de desarrollo comenzó a tomar forma, anticipando la llegada de la tarjeta controladora a utilizar.

A continuación, se muestra la tarjeta (figura 2.8) con la que trabajo en la primera fase del desarrollo.

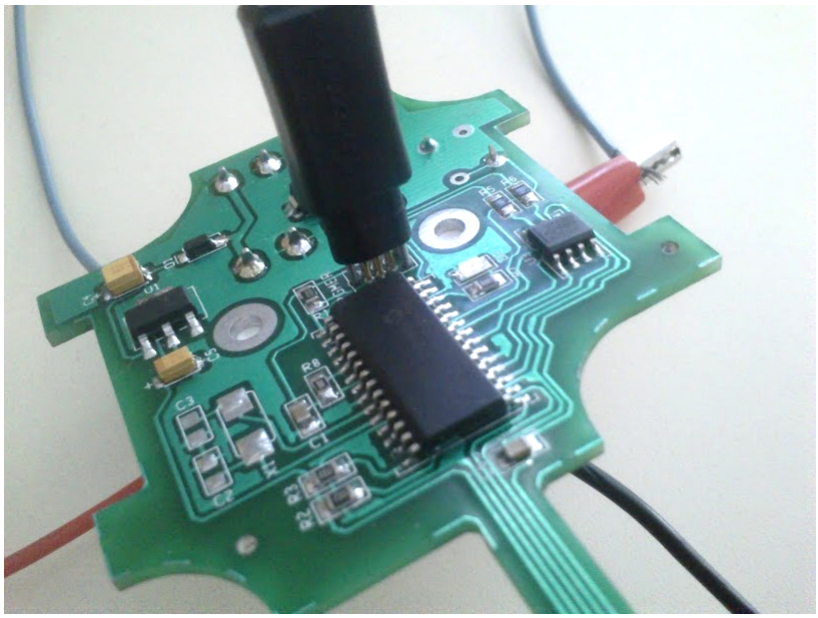

Figura 2.8: Tarjeta de pruebas y aprendizaje

En este punto del desarrollo, lo más importante que se debía considerar, era el microcontrolador utilizado, y ya que tanto la tarjeta de pruebas como la tarjeta final compartían el mismo microcontrolador, fue posible ir avanzando en el desarrollo del firmware.

El firmware comenzó a ser desarrollado únicamente con los requerimientos y funcionalidades que debía cumplir, simulando que controlaba distintos periféricos, tal como debería hacerlo en su versión final.

Dicho proceso fue de vital importancia para tener la versión casi final, ya que se pudieron anticipar diversos casos en la programación, todo con la finalidad de que cuando este firmware se programe en el producto final, únicamente se le harían adaptaciones mínimas y pruebas de integración.

La programación de ambas tarjetas, la de pruebas y la versión final, se realizaba mediante una interfaz de programación llamada tag-connect (figura 2.9). La cual cuenta de 6 puntos de conexión, asociados a ciertos pines del microcontrolador encargados de realizar la carga del firmware. Ademas del cable y su respectivo conector, también se debe utilizar una herramienta capaz de comunicar el software utilizado para codificar el firmware con la tarjeta controladora.

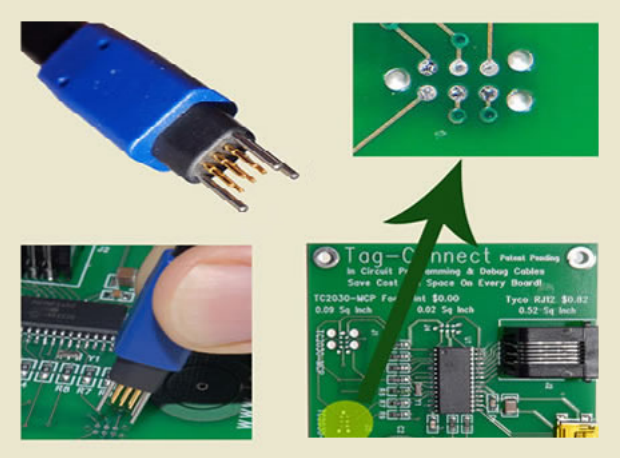

Figura 2.9: Se muestra cable e interfaz Tag-connect, tomada de [22]

La herramienta encargada de cargar el firmware en la tarjeta es conocida como programador, en este caso se utilizó el siguiente modelo, el Pickit 3 (figura 2.10), fácil de utilizar, portátil y poseedor de comunicación USB. Este programador posee características como la de leer, escribir y borrar información de un microcontrolador, además de ser multiplataforma y soportar una amplia gama de estos.

Figura 2.10: Programador Pickit 3

2.7.2. Software y compilador utilizados

El software final utilizado para codificar el firmware de la tarjeta fue MPLAB® X Integrated Development Environment (IDE) en su versión 3.40. Un software multiplataforma de la compañía Microchip, el cual, según su página oficial lo describe como una interfaz de desarrollo porque proporciona un "entorno" integrado único para desarrollar código para microcontroladores, además de que ya se contaba con el programador de la misma compañía, así que las opciones eran escasas [16].

Sin embargo, en el párrafo anterior se menciona "el software final", pero ¿por qué se hace tanto énfasis en ese aspecto? La respuesta es sencilla, ya que inicialmente se empezó a trabajar por recomendación con otro entorno de desarrollo llamado PIC C Compiler. Sin embargo y a pesar de jactarse de ser el primer entorno de desarrollo concebido hace más de 20 años para microcontroladores Microchip, su interfaz no era nada intuitiva pero funcional. Aunque cuando se buscó utilizar el programador Pickit 3 con el software, la carga de firmware al microcontrolador no se realizó de la manera esperada. Lo importante de este IDE, más que su interfaz, eran las bibliotecas de desarrollo que proporcionaban un conjunto de instrucciones bastante funcionales que harían el desarrollo más ágil y eficiente.

Ante la incertidumbre sobre qué debería ocuparse, finalmente logró solucionarse el problema de la siguiente forma: utilizar las microinstrucciones que PIC C Compiler proveía en MPLAB® X Integrated Development Environment (IDE) (figura 2.11), de esa forma el programador funcionaria sin inconveniente alguno.

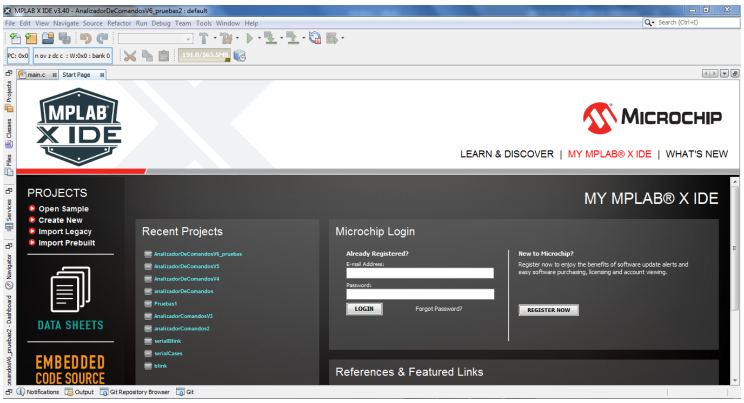

Figura 2.11: Pantalla inicial del IDE MPLAB® X Integrated Development Environment (IDE)

Para integrar el compilador a MPLAB se llevaron a cabo los siguientes pasos una vez abierto el entorno de desarrollo:

- Visitar la pestaña en la barra superior de herramientas, llamada "Tools" y buscar la sección "Plugins".
- En la nueva sección, seleccionar la pestaña "Available plugins" y buscar en la barra de búsqueda "CCS C Compiler".
- Seleccionar en el recuerdo correspondiente, descargarlo e instalarlo.
- Ahora aparecerá listado en los complementos instalados (figura 2.12).

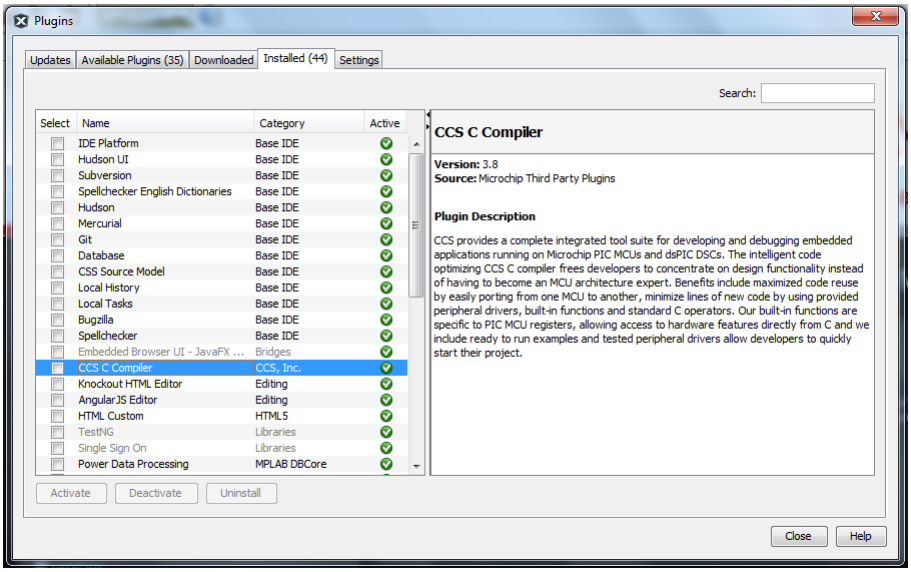

Figura 2.12: Vista del compilador instalado

Para utilizar el compilador, será necesario crear un proyecto nuevo e indicar qué compilador se utilizará. Al realizar dicha acción ya se encuentran asociadas las instrucciones de dicho compilador a nuestro entorno de desarrollo (figura 2.13).

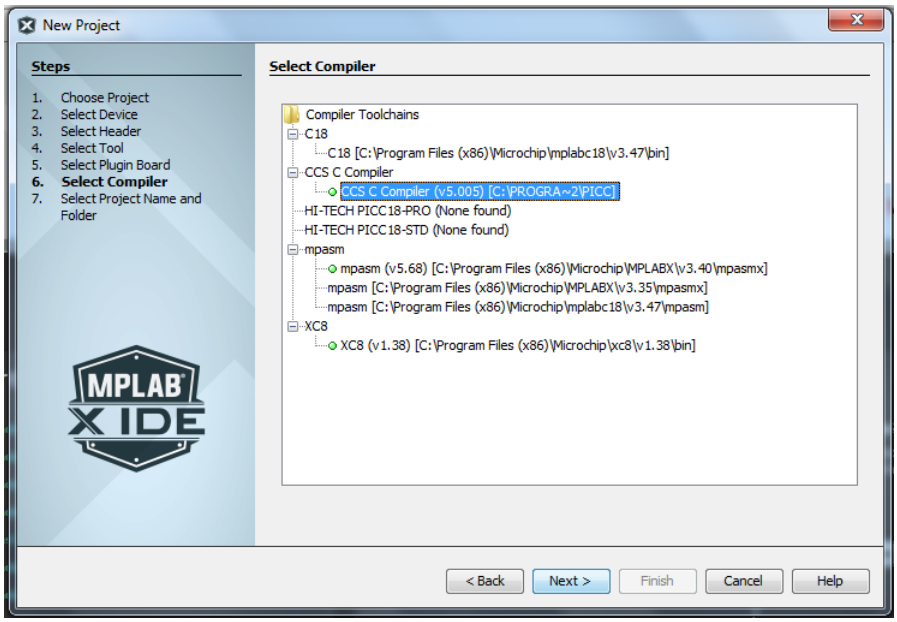

Figura 2.13: Selección del compilador al crear nuevo proyecto

Cuando se desea llevar a cabo la carga del firmware programado en el entorno de desarrollo, basta con conectar el programador a la computadora vía USB en este caso, y seleccionar el símbolo de ejecutar programa, el cual se encuentra en la barra de herramientas en la parte superior (figura 2.14).

Figura 2.14: Detalle de la barra de herramienta; con círculo en rojo símbolo de ejecución de programa

Una vez configurado el software con su respectivo compilador, se puede proceder a llevar a cabo la utilización del mismo las veces que sea necesario, ya que el microcontrolador se programa instantáneamente y no se requiere de algún instrumento especial para hacerlo funcionar. Cabe mencionar que el programador Pickit 3, no suministra la alimentación necesaria al microcontrolador por lo que se requiere tener energizada la tarjeta controladora mediante una fuente de corriente directa.

2.7.3. Diagrama de identificación de los periféricos de la tarjeta controladora

A continuación, se muestra el diagrama físico de periféricos (figura 2.15) generado a partir del diagrama eléctrico (que se presentará en el capítulo siguiente) proporcionado por la coordinación de electrónica del Instituto de Ingeniería. En el cual se identifican los componentes de la tarjeta y el nombre de estos.

La utilidad de este diagrama fue para identificar los periféricos del microcontrolador y sus respectivos números de parte, para así poder obtener información de cada uno y lograr una interacción satisfactoria entre todos los periféricos y el microcontrolador.

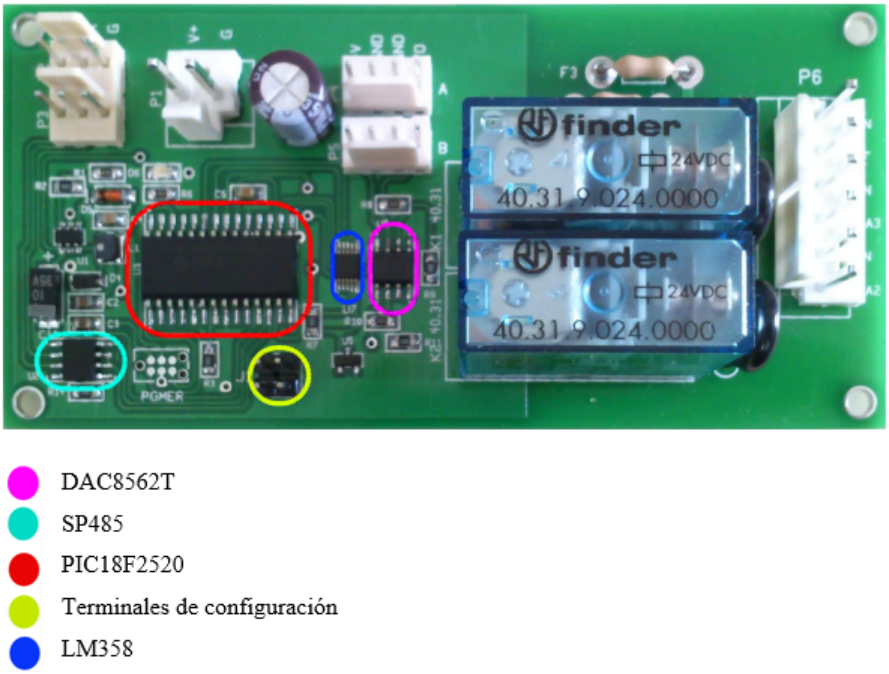

Figura 2.15: Identificación de los componentes de la tarjeta controladora

En la figura 2.15 se observan los periféricos que se explicarán a continuación, ya que son parte importante en el desarrollo del firmware y la interacción con el software que la tarjeta controladora debe desempeñar.

2.7.4. Periféricos internos

A continuación se presenta la sección encargada de explicar y describir los periféricos internos con los que cuenta la tarjeta controladora, previamente identificados en la figura 2.15.

2.7.4.1. PIC18F2520

Es la unidad principal que controla todos los periféricos, ya se internos o externos. Dependiendo el modo en el que se utilice y como sea configurado, actuará como gestor de conexiones, intérprete de comandos y finalmente como controlador de los periféricos que lo acompañan.

La funcionalidad principal del microcontrolador (figura 2.16) en esta aplicación es la de interpretar un comando recibido y responder de acuerdo con lo establecido en la programación, se podría decir que es el encargado de administrar la comunicación entre la computadora y los periféricos con los que cuenta.

Todo el código utilizado en la programación de la tarjeta fue pensado para aprovecharla al máximo y desempeñar de manera eficiente todas sus tareas. Así mismo para cumplir con los requerimientos solicitados para elaborar el sistema deseado.

Principales características del PIC18F2520 según el fabricante [17]

- Versión mejorada del 18F252
- Arquitectura de 8 bits
- Memoria flash: 32 Kbytes
- Memoria SRAM: 1536 bytes
- Memoria EEPROM: 256 bytes
- Convertidor analógico-digital: 10 bit
- Puertos PWM: 2
- Protocolos de comunicación: SPI e I2C
- Frecuencia de operación: 40 MHz

Figura 2.16: Microcontrolador de una tarjeta controladora visto bajo un microscopio

2.7.4.2. DAC8562T

Encargado de controlar las servo válvulas utilizadas para variar la presión de la cámara triaxial. El DAC (figura 2.17) se encarga de convertir los impulsos suministrados de forma digital por el microcontrolador a su equivalente en una señal analógica.

La forma en que se comunican los dispositivos se lleva a cabo de manera unidireccional, por ser únicamente el microcontrolador el que transmite al DAC, nunca se espera una confirmación del DAC. Sin embargo, como la comunicación se lleva mediante el protocolo SPI, se tiene una comunicación síncrona con los pulsos de reloj, el cual controlará el modo en el que se enviarán lo datos. Este protocolo posee una velocidad de transmisión mayor, comparada con protocolos como i2c o SMBus [18].

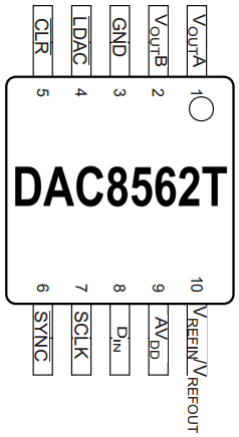

Figura 2.17: Esquema de conexión del DAC utilizado en la tarjeta, recuperado de [1]

2.7.4.3. LM358-N

Se trata de un amplificador operacional encargado de amplificar una señal de voltaje. El amplificador cuenta con un gran rango de voltajes de alimentación, los cuales pueden ir de los 3[V] a los 32[V] en corriente directa. De igual forma se puede utilizar una fuente simétrica en el rango de 1.5[V] hasta los 16[V].

El LM358-N (figura 2.18) es el circuito encargado de controlar las bobinas de los relevadores, ya que no hay una manera de realizarlo directamente con el microcontrolador y en caso de intentarlo, la alta demanda de corriente por parte del relevador provocaría desperfectos en la tarjeta.

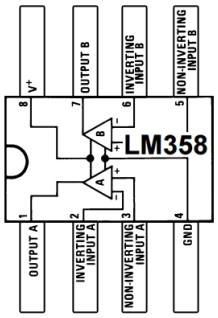

Figura 2.18: Diagrama de conexiones del circuito LM358, recuperado de [2]

2.7.4.4. SP485

Es el circuito integrado (figura 2.19) encargado de realizar la comunicación mediante el protocolo RS-485. Su diseño le permite una operación de bajo consumo eléctrico sin sacrificar desempeño. Sus pines de conexión son mínimos y la electrónica necesaria para hacerlo funcional es escasa, ya que el circuito está prácticamente listo para funcionar.

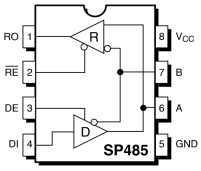

Figura 2.19: Diagrama de conexiones del integrado SP485, recuperado de [3]

2.7.4.5. Relevadores

Un relevador es un dispositivo electromecánico, encargado de actuar mediante un campo electromagnético inducido por una bobina. Su utilidad es la de activar dispositivos de mayor potencia que el de entrada. Para el circuito de la tarjeta controladora, el relevador (figura 2.20) cumple la función de energizar las válvulas de cierre y apertura del sistema de aire.

Figura 2.20: Vista superior de los relevadores en la tarjeta controladora

2.7.5. Periféricos externos

Una vez que se explicó a grandes rasgos cuáles son los elementos que forman parte de la tarjeta controladora, es turno de hablar un poco de los periféricos externos, los cuales son aquellos a los que se controlará por medio de la tarjeta. En los capítulos subsecuentes se explicará como funciona cada periférico, y como es que se lleva a cabo la interacción entre los periféricos internos y los periféricos externos.

2.7.5.1. Servo válvulas

Estas son utilizadas en el sistema para control de presión. El modelo utilizado cuenta con una configuración especial, que le permite variar la presión dependiendo el voltaje que le sea suministrado. En este caso, es el DAC quien se encarga de escribir valores en las válvulas (figura 2.21) escribiendo el voltaje que indique previamente el controlador, de esta manera es posible variar de manera efectiva la presión mediante un dispositivo electrónico.

Figura 2.21: Servo válvula de prueba conectada al sistema de aire

2.7.5.2. Válvula check

La utilización de los relevadores se lleva a cabo de manera independiente, dado que sirven para controlar diversos mecanismos. Cada relevador (figura 2.22) se encuentra conectado a una válvula, la cual permite o impide el paso de un fluido, el cual en este caso se trata de aire a cierta presión. Así prácticamente todos los periféricos trabajan en conjunto con un solo fin, el de controlar la cámara triaxial de manera digital.

Figura 2.22: Cada relevador controla una válvula

2.8. Esquema de funcionamiento de la tarjeta

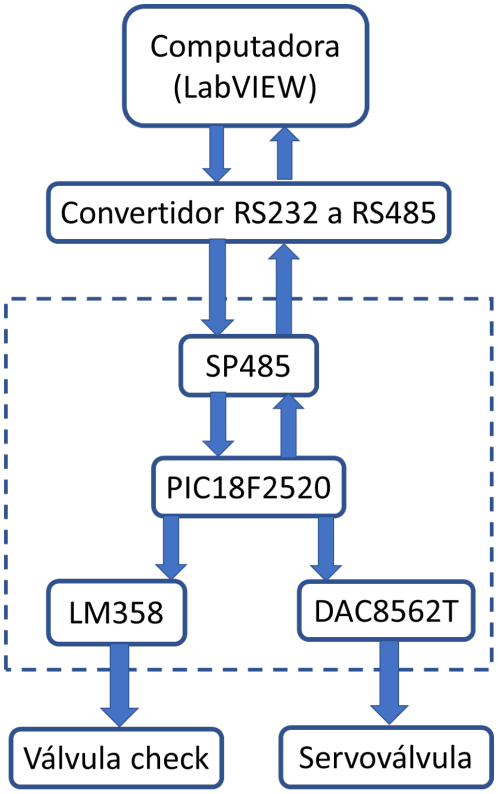

Figura 2.23: Esquema de funcionamiento

En la figura 2.23 los componentes que se encuentran dentro de la linea punteada, representan en conjunto a la tarjeta controladora, únicamente se muestran aquellos que interactúan para llevar a cabo las tareas para las cuales fue diseñada.

El funcionamiento de la tarjeta da inicio cuando por medio de un software elaborado en LabVIEW, ejecutándose en una computadora, se envía algún comando por medio de un puerto USB de la computadora asociado a la comunicación RS232, el cual con la ayuda de un convertidor RS232 a RS485, llega al chip encargado de la comunicación con la tarjeta, el SP485. A su vez, éste se comunica con el microcontrolador de la tarjeta (PIC18F2520), cuyo comando recibido se interpretará y ejecutará. Por ejemplo, si se trata de un comando para cerrar las válvulas, el PIC escribe un valor para activar o desactivar la válvula correspondiente; o bien, en el caso que se busque escribir un valor en la servoválvula, el PIC escribe los valores necesarios para

que el DAC8562T controle la servoválvula. Tras haber ejecutado cualquiera de ambos comandos, el PIC responde a la computadora con un "OK". Cabe destacar que esta última acción no se encontrará disponible en la versión productiva del sistema, dado que los mensajes de respuesta podrían saturar el canal de comunicaciones.

Capítulo III

Firmware

Para poder abordar este capítulo desde una mejor perspectiva, es preciso iniciar definiendo: ¿qué es un firmware? y ¿por qué usamos un firmware?

Según el Institute of Electrical and Electronics Engineers [11, 1990, pp. 33], un firmware se define como la combinación de instrucciones de un dispositivo de hardware e instrucciones y datos de computadora que residen como software de solo lectura en el mismo. Dada esta definición, podemos concluir que se trata de un programa inalterable, el cual trabajará al menor nivel lógico que sea posible, además de que se encargará de controlar los circuitos electrónicos acorde a su programación establecida. En resumen, el firmware es un software encargado de controlar al hardware.

Cabe destacar, que cualquier mejora que se quiera realizar al firmware establecido en un dispositivo, deberá actualizarse por medio de una interfaz especial, provocando que la tarea sea de manera rudimentaria además de que cualquier fallo que pudiera presentarse en la actualización del mismo, podría provocar un daño irremediable en el dispositivo que lo contiene; un fallo muy común que se presenta al actualizar un firmware es el de alimentación, ya que si por algún motivo, en medio de un proceso de actualización se corta el suministro de energía, la carga del nuevo firmware quedaría inconclusa, dañando parte del firmware actual y parte del especializado para realizar las cargas de actualizaciones.

3.1. Proceso de configuración

Llamaremos proceso de configuración a la serie de pasos iniciales previos a la puesta en marcha del sistema, el cual deberá de contemplar ciertas configuraciones para asegurar un funcionamiento ideal y que en el momento que todo se encuentre correctamente conectado, el sistema pueda funcionar con normalidad.

Afortunadamente las configuraciones a considerar son realmente pocas, algunas son referentes a los periféricos que se controlarán, otros relativos a la tarjeta por sí misma y finalmente los elementos externos (alimentación eléctrica) que hacen posible la puesta en marcha del sistema.

Dado que el proceso de configuración para la puesta en marcha de la tarjeta es realmente importante, en los siguientes subtemas se ilustrará la manera correcta de realizar la identificación de los periféricos principales, así como las conexiones necesarias de energía y comunicación.

3.1.1. Conexiones de la tarjeta controladora

Para el desarrollo del firmware fue proporcionado un esquema (figura 3.1) de las principales conexiones internas de la tarjeta. En él se especifican con que periféricos cuenta, hacia donde se conectan y con quienes interactúa.

En el esquema de la página podemos ver los periféricos internos, a la izquierda y nombrado MAX485 se encuentra el convertidor RS485, encargado de recibir comandos y enviar respuestas.

En la parte central se ubica el microcontrolador y es el que concentra todos los periféricos, encargado de comunicar y gestionar las comunicaciones entre periféricos. Inmediatamente conectado al microcontrolador mediante tres pines especiales, se halla el DAC8532, encargado de convertir un impulso digital en una señal analógico entendible por la servo válvula.

Cabe destacar que dicho esquema se proporcionó por parte del creador de la tarjeta de desarrollo, sirvió para identificar los componentes, número de parte y demás periféricos que se utilizaron en esta. Sin duda fue de gran utilidad para conocer a fondo la tecnología con la que se trabajaría.

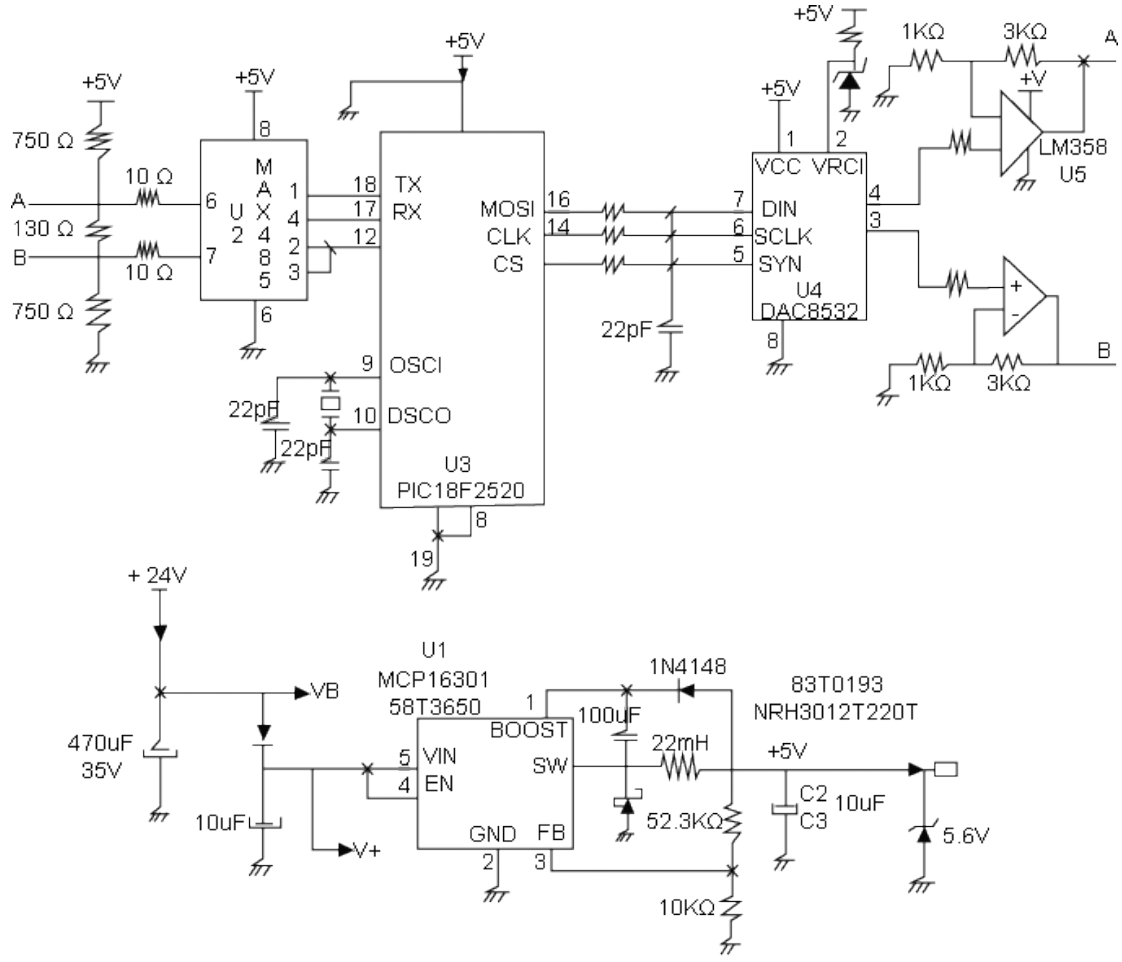

Figura 3.1: Esquema proporcionado por la coordinación de electrónica

En la siguiente figura 3.2 se muestra la tarjeta físicamente.

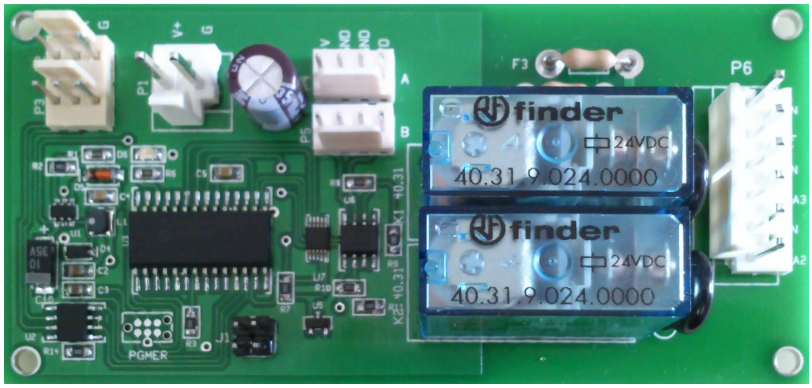

Figura 3.2: Tarjeta física

Al lado izquierdo se encuentra la interconexión con otras tarjetas mediante el protocolo RS485, se trata de dos conectores, donde uno debe conectarse a la computadora y el otro servirá para crear un nuevo canal de transmisión, haciendo posible la intercomunicación con una tarjeta más.

A su derecha se pueden distinguir dos pines que sirven para proveer la alimentación de 12 [V] necesaria para hacer funcionar la tarjeta.

Casi al centro de la tarjeta, se pueden hallar dos conectores de cuatro pines cada uno, dichos conectores sirven para controlar la servo válvula encargada de variar la presión. Cada terminal se encarga de controlar una servo válvula distinta.

Finalmente, en el extremo derecho se distingue un gran borne, el cual es utilizado para controlar las dos válvulas de propósito general, encargadas de abrir y cerrar electrónicamente. Es decir, cada válvula permitirá dejar o no pasar determinado fluido a través de sí. En este caso, se trata de aire. Cabe mencionar que estas válvulas no regulan presión, únicamente se dedican a impedir el paso del aire o permitir que fluya a través de ella cuando se le indique.

3.1.2. Configuración inicial de la tarjeta controladora

Dado que el firmware de la tarjeta controladora tuvo que adaptarse a su diseño, cuenta con características peculiares que hacen posible una autonomía por parte de ésta, la cual requiere escaso apoyo humano para ser configurada y utilizada.

En esta sección, se explicará cómo es que trabaja el firmware de la tarjeta y la manera como se autoconfigura modificando algunas entradas de esta.

El paso inicial para configurar una o más tarjetas, consiste en programarlas. Esto se hace mediante una herramienta de hardware especializada y un software encargado de cargar el código al microcontrolador.

La versatilidad del código desarrollado para estas tarjetas lleva consigo una autonomía muy importante, ya que el código que se carga a cada tarjeta es el mismo, sin embargo, la configuración especial que tiene cada tarjeta hace que se comporte de manera diferente. Esto es posible gracias a un par de pines con los que cuenta cada tarjeta, se podría decir que son los pines iniciales de configuración y al ser pines físicos, no se requiere de una reprogramación de la tarjeta para poder utilizarlos.

Cada tarjeta en sí tiene un número de asignación diferente definido al configurarla mediante sus pines de configuración (figura 3.3), lo cual provoca que, aunque los comando que lleguen a recibir tengan una estructura parecidas, estas descartaran todos aquellos comandos que no les correspondan.

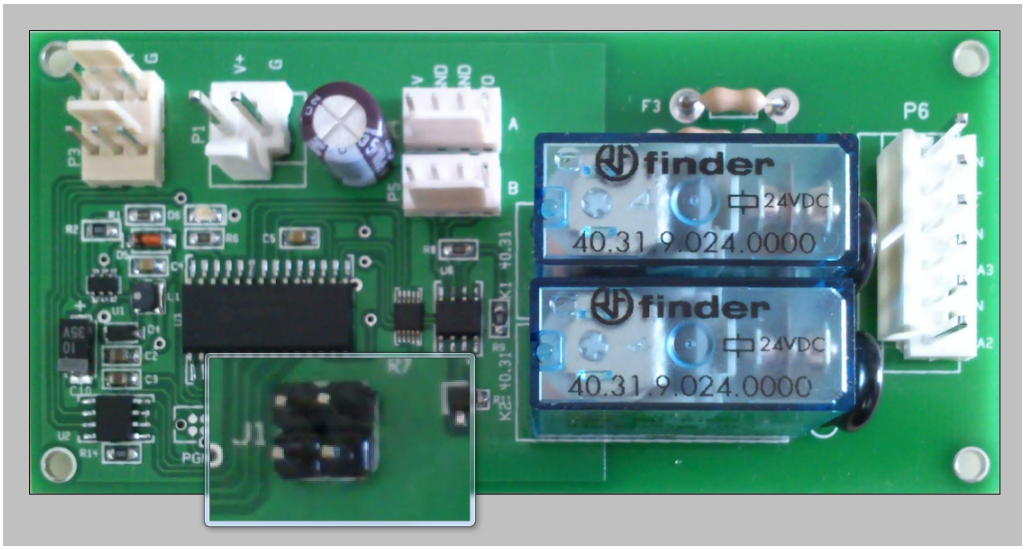

Figura 3.3: Detalle de la ubicación de los pines de configuración

Se puede decir que es un pequeño tablero donde podemos seleccionar que combinaciones de números queremos utilizar, dada la naturaleza de la tarjeta únicamente podemos elegir entre valores de 0 [v] y 5 [v], es decir que tenemos únicamente valores lógicos de 0 y 1.

Para energizar correctamente cada pin de configuración, debemos utilizar un jumper y así cada pin reciba el valor lógico esperado. En este caso se cuenta con cuatro diferentes configuraciones, las cuales se muestran en la figura 3.4:

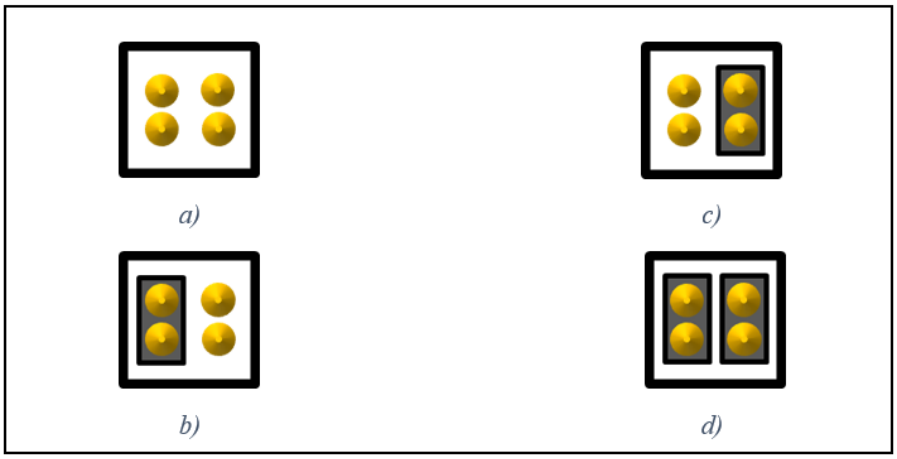

Figura 3.4: a) Dirección 0 b) Dirección 1 c) Dirección 2 d) Dirección 3

Cualquiera de las combinaciones anteriores es válida para iniciar con la puesta en marcha del sistema, ya que la subrutina de configuración del microcontrolador lleva a cabo al iniciarse, la configuración de que dirección le será asignada, todo con la finalidad de recibir únicamente la información que le corresponda y actué de manera autónoma. La parte de código encargada de verificar la configuración y asignar la dirección que le corresponde a cada dispositivo es la siguiente:

```
char addressDisp ='0';
if (input_state(PIN_A0))
        addressDisp = '0'+1;
if (input_state(PIN_A1))
        addressDisp +=2;
```

Código 3.1: Configuración del número de dispositivo

En esta parte se verifican los jumpers de dirección del dispositivo, existen 3 casos como lo explicamos previamente, el caso que llamaremos DEFAULT, donde no se cumplen ninguna de las siguientes condiciones, es decir que ningún pin se encuentra energizado o conectado, entonces la variable no tiene que modificarse y se mantiene

como “0”, valor que será la nueva dirección del dispositivo, de encontrarse el pin A0 en un estado alto, la dirección del dispositivo será sobrescrita y ahora tendrá la dirección 1 (siempre y cuando A1 sea 0), ya que si A0 se encuentra en estado bajo y A1 en estado alto, la nueva dirección del dispositivo será 2.

Finalmente, y para tener la última dirección posible en nuestras tarjetas, deberemos tener tanto A0 como A1 en estado alto, provocando que las dos instrucciones se cumplan y la dirección se modifique en dos ocasiones, primero siendo temporalmente la dirección 1 pero enseguida en la línea contigua de código acumulándose para valer finalmente 3. Esta configuración asignaría como número de dispositivo el 3.

3.2. Proceso de recuperación

Según Wikipedia [15], en el ámbito de la memoria, el proceso de recuperación o recuerdo consiste en la evocación de sucesos, eventos o información almacenada en el pasado. Tal es el caso de nuestro sistema, el cual por sí mismo no sería capaz de recuperarse tras algún fallo, ya sea de alimentación o mecánico. En este caso se elaboró una pequeña subrutina de código, la cual se encarga de ir guardando en memoria el ultimo valor o estado conocido del programa, por si llegara a existir un fallo, iniciar donde se quedó.

El sistema de recuperación trabaja en conjunto con la memoria no volátil que contiene el microcontrolador, lo que le permite retomar las pruebas (las cuales, por la información proporcionado por encargados del laboratorio de geotecnia, llegan a demorar entre 3 y 15 días, dependiendo el tipo de prueba) en cierto punto para que se cuente con algo de información sin haber desperdiciado tiempo en la prueba.

3.2.1. Secuencia para retomar pruebas

Las pruebas se retoman al iniciar el programa, si no llegara a haber algún fallo, los valores iniciales que tendrá guardados el microcontrolador deberían ser equivalentes a 0, para así asegurar que, al comenzar una nueva prueba, tengamos valores de configuración.

Esta comprobación y configuración se lleva a cabo en las siguientes líneas de código, las cuales a su vez mandan a llamar a otra subrutina encargada de lo propio.

```
escribe_DAC(MODEA,read_EEPROM(dacLowA_res),read_EEPROM(dacHighA_res));
escribe_DAC(MODEB,read_EEPROM(dacLowB_res),read_EEPROM(dacHighB_res));
```

Código 3.2: Llamando a la subrutina de escritura

Las dos anteriores líneas de código ejecutan la subrutina de escritura en el DAC, la cual recibe como parámetros los siguientes valores, el primer parámetro se trata en ambos casos de un número expresado en formato binario, y dependiendo el caso podría ser el número 0b00011000 (MODEA) o 0b00011001 (MODEB), el cual representa el DAC que va a funcionar, ya sea el A o el B. Los últimos dos parámetros que recibe la función son muy parecidos en la forma en que se leen, ya que en ambos casos se utiliza la opción de lectura de la memoria EEPROM que tiene el microcontrolador. Inmediatamente los valores se envían a la subrutina y se escriben los valores de respaldo en el DAC.

Prácticamente la mayor parte de instrucciones son para la configuración y utilización del DAC, es decir para hacerlo funcionar en modo escritura. A su vez se utilizó una pequeña espera de 5 [μs] para el correcto funcionamiento de envió de valores. Después de eso se utiliza una microinstrucción, la cual escribe en una memoria interna los datos recibidos en el comando de escritura, en una sección del programa que se encargará de conservar dichos valores para cuando se requiera utilizarlos, puede ser si se presenta un fallo en la alimentación y se desea retomar el trabajo previamente llevado a cabo, finalmente se quita el modo de escritura del DAC para prevenir cambios en los valores escritos.

En la siguiente página, se ilustra por completo la subrutina encargada de llevar a cabo la escritura en los DAC.

```
void escribe_DAC(int mode, int dacLow, int dacHigh)
{
        output_high(PIN_B0); //SYNC = 1 deshabilita registro
        output_low(PIN_B2); //LDAC
        output_high(PIN_B1); //CLR
        output_low(PIN_B0);  //SYNC = 0 habilita registro
        delay_us(5);
        spi_write(mode);
        spi_write(dacHigh);
        spi_write(dacLow);
        output_high(PIN_B0); //SYNC=1 deshabilita registro
        output_low(PIN_B2); //LDAC

        /* COMIENZA RESPALDO DE VALORES FINALES DAC */
        if(mode == MODEA)
        {
                write_eeprom(dacHighA_res,dacHigh);
                write_eeprom(dacLowA_res,dacLow);
        }
        else if(mode == MODEB)
        {
                write_eeprom(dacHighB_res,dacHigh);
                write_eeprom(dacLowB_res,dacLow);
        }
        /*FINALIZA EL RESPALDO DE VALORES FINALES DAC*/
}
```

Código 3.3: Subrutina de escritura en DAC

3.2.2. Valores de recuperación

Los valores que son considerados como valores de recuperación, mencionados previamente, son aquellos almacenados en la memoria interna del microcontrolador. Estos se almacenan de manera secuencial y forzada, que quiere decir esto, que se reservaron localidades adyacentes y especiales para llevar a cabo su almacenamiento.

Dicho almacenamiento fue limitado por los valores máximos que iban a almacenarse y considerando las capacidades del microcontrolador. Únicamente se utilizan cuatro localidades de memoria, de la siguiente manera (figura 3.5):

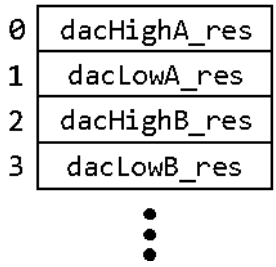

Figura 3.5: Mapa de memoria valores de respaldo

```
/*DIRECCIONES DE RESPALDO EEPROM*/
#define dacHighA_res 0
#define dacLowA_res 1
#define dacHighB_res 2
#define dacLowB_res 3
```

Código 3.4: Definiendo localidades de valores de respaldo

Siendo las localidades 0 y 1, las encargadas de almacenar los valores de respaldo correspondientes al DAC-A, mientras que las localidades 2 y 3 se encargan de guardar lo referente al DAC-B.

3.3. Proceso de lectura e interpretación de comandos

Por la naturaleza misma del sistema y por cómo fue diseñado, se optó por la utilización e implementación de comandos lo más sencillos posibles, compactos y

con la información mínima para evitar confusión a la hora de leerlos, codificarlos y ejecutarlos.

Para la elaboración de los comandos del sistema se siguieron las normas listadas a continuación para la adaptación de los comandos y asegurar un funcionamiento lo más sencillo posible, es decir que se tenía bien claro que acciones se llevarían a cabo mediante cada comando.

1. Cada comando deberá de iniciar con los caracteres “G” y “R”.

2. La longitud máxima del comando deberá ser tomada del comando más largo, si existiese algún comando de mayor longitud tendrá que ser descartado totalmente, ya que el canal de comunicaciones no debe almacenar ningún buffer.

3. El comando contendrá la dirección del dispositivo a controlar, la versión actual de hardware y software permiten únicamente un máximo de 3 dispositivos por lo descrito en la sección de configuración del dispositivo.

4. Si se ingresa una dirección de algún dispositivo que no está conectado o no existe, no habrá respuesta.

5. En caso de querer controlar alguna servo válvula se tendrá que indicar mediante el carácter “V” precedido de “0” o “1” para indicar de que servo válvula se trata, inmediatamente se agrega el carácter “0” o “1”, donde “0” significa apagar y “1” encender.

6. Para hacer la utilización del DAC el comando deberá tener en cuenta las consideraciones de los puntos 1,2 y 3, enseguida añadir el carácter “D” haciendo referencia a la utilización del DAC, después añadir “A” o “B” para especificar de qué dispositivo se trata (ver esquema de conexiones), finalmente añadir los cuatro caracteres correspondientes al valor a escribir en el DAC correspondiente. Veáse la tabla 3.1.

7. Todos los comandos que busquen ser validos deberán contener al final de ellos los caracteres de control “\r”(carriage return) y “\n” (newline) sin excepción.

MSB															LSB	DEC	HEX
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	255	FF
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	65280	FF00
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	65535	FFFF

Tabla 3.1: Valores y equivalentes

El programa en todo momento se encuentra escuchando mediante un bus de transferencia de datos, a excepción de que se especifique que enviara alguna respuesta de ejecución. Los mensajes viajan de una computadora al microcontrolador y en algunas ocasiones el flujo se ve modificado, siempre y cuando el microcontrolador tenga respuesta de algún proceso ejecutado. Si el comando enviado no está completo o carece de consistencia en su estructura, se descarta y se sigue a la escucha de nuevos comandos.

Una vez que se cuenta con un comando al que podríamos clasificar como comando bien formado, es momento de interpretarlo a modo de que el microcontrolador sea capaz de proveer una respuesta y llevar a cabo su ejecución. A este proceso lo llamaremos interpretación, la cual se hace de manera muy elemental y haciendo uso de las bibliotecas estándar para el microcontrolador.

Esta interpretación es ejecutada en un bucle continuo, que se encarga de inicializar el comando con valor de '\0', después de eso, el microcontrolador deberá entrar en modo de recepción de datos, y por la lógica del protocolo RS485, se prosigue a desactivar la salida de datos, todo con la finalidad de evitar alguna colisión entre datos de entrada con los de salida si es que llegara a existir alguno de ellos. Con esto aseguramos un canal unidireccional, capaz de únicamente recibir comandos.

Mientras no enviemos los caracteres de control ya mencionados, no serán tomados en cuenta los datos recibidos en el bus de comunicación. Para verificar eso, se lee carácter a carácter y se asigna a una variable auxiliar encargada de ir formando el comando correcto.

Una vez que se tiene un comando bien con los caracteres de control, se prosigue a verificar que sea un comando válido, esto se hace en primera instancia validando que tenga una longitud de al menos cinco caracteres y máximo once, si esto se cumple cada vez se tiene mayor certeza de que el comando leído podría ser correcto. Como

únicamente se tienen un par de comandos válidos, únicamente se tienen dos casos de revisión, para el comando de servo válvulas y el de escritura en el DAC. Se toma como dos comandos, ya que la estructura es de la misma longitud independientemente de si se requiere controlar el dispositivo A o B.

Para que los comandos se ejecuten, únicamente se valida que cada comando tiene los caracteres que debe tener en la posición deseada, por ejemplo, que inicien con los caracteres "G", en la segunda posición exista el carácter "R", que el tercer carácter corresponda al número de dispositivo, seguido del identificador de "V" (válvula) o "D" (DAC), finalizando con los argumentos respectivos para cada caso.

Para cualquier comando con longitud correcta y caracteres de finalización, pero con valores incorrectos, se descarta y se reinicia la memoria utilizada para guardar el comando.

3.3.1. Proceso de ejecución de comandos

Este proceso se lleva únicamente cuando se ha discernido que es un comando prácticamente listo para ser ejecutado. Dicho proceso se da ya que previamente se descartaron los mensajes incorrectos y se filtraron los mensajes correctos con la finalidad de llevar a cabo su ejecución.

Si se busca encender o apagar alguna servo válvula, se hace uso de una subrutina llamada "enciende_valvula" la cual recibe dos parámetros, el parámetro uno encargado de seleccionar la válvula a operar, el segundo parámetro es el encargado de encender o apagar la válvula.

Sin importar que operación se lleve a cabo, se responderá al dispositivo de salida el estado de ejecución, indicando en que dispositivo se ejecutó y un mensaje personalizado.

En la siguiente página se muestra el código de la subrutina correspondiente para el control de las válvulas.

```
void enciende_valvula(char A, char B){
        if(A =='0')
        {
                switch(B){
                        case '0':
                                output_low(PIN_A2);
                                msjEstado(0);
                        break;
                        case '1':
                                output_high(PIN_A2);
                                msjEstado(0);
                        break;
                }
        }
        else if(A =='1')
        {
                switch(B){
                        case '0':
                                output_low(PIN_A3);
                                msjEstado(0);
                        break;
                        case '1':
                                output_high(PIN_A3);
                                msjEstado(0);
                        break;
                }
        }
}
```

Código 3.5: Subrutina para manejo de apertura y cierre de válvulas

La subrutina anterior es la encargada de la apertura y cierre de los dos relevadores que se encuentran en la tarjeta controladora, los cuales más adelante se encargarán de controlar dos válvulas, ya sea para dejar o no pasar aire para suministrar presión. Esta subrutina únicamente se ejecutará cuando el comando contenga la siguiente forma: GR#V##.

Donde como ya se explicó previamente GR son dos constantes que especifican que se trata de dicho grupo de tarjetas controladoras, el primer símbolo de # se reemplaza por el número de tarjeta controladora generado por los pines de configuración y los dos símbolos contiguos de ## hacen referencia a las distintas configuración de apertura y cierre de válvula, ya que en el primer caso se puede elegir la válvula 0 con la acción de abrir (asociada al número 1) generando el comando GR0V01, en caso de querer abrir la válvula 1 el comando quedaría formado de la siguiente forma GR0V11.

Si se busca hacer uso de un DAC, y cuando se tiene un mensaje para este fin, la primera operación que se lleva a cabo es la de dividir y asignar las diferentes partes del comando para poder ser utilizadas de una manera sencilla y eficiente en el proceso de ejecución.

Inmediatamente y ya con el comando separado, se prosigue a escribir en el DAC seleccionado el valor indicado.

Cabe destacar que en la "separación" del comando también se convierte a su equivalente respectivo en hexadecimal (base 16) cada parte de lo que fue separado. Esta operación se realiza gracias a una función que provee el compilador, llamada stroul, encargada de convertir una cadena (la leída) a un entero sin signo y dependiendo de base requerida, es la que se especifica.

Para separar el comando y poderlo interpretar más adelante, se utilizó un arreglo auxiliar estático de 3 caracteres, se inicializó su porción de memoria e inmediatamente se realiza la copia del comando recibido en los arreglos de caracteres auxiliares. Finalmente se convierte a base 16 el valor de cada arreglo auxiliar y se asigna al valor a escribir en el DAC.

```
void recibeCadena(char *CMD)
{
        char aux1[3];
        char aux2[3];
        memset(aux1, '\0', sizeof(aux1));
        memset(aux2, '\0', sizeof(aux2));
        // aux1 = { \0,\0,\0,\0,\0,\0,\0,\0,\0,\0 };
        // aux2 = { \0,\0,\0,\0,\0,\0,\0,\0,\0,\0 };
        strncpy(aux2, CMD + index, 2);
        aux2[2] = '\0';
        strncpy(aux1, CMD + index+2, 2);
        aux1[2] = '\0';
        dacHigh = strtoul(aux2,NULL,16);
        dacLow  = strtoul(aux1,NULL,16);
}
```

Código 3.6: Sección de tratamiento de los comandos del DAC

3.3.2. Proceso de escritura en periféricos

Previamente se analizaron los periféricos involucrados en este desarrollo y se ahondo en las principales características de cada uno, con la finalidad de unir la parte física y la lógica del código.

El proceso de escritura en los periféricos se realiza de manera muy sencilla gracias a las herramientas seleccionadas para programar el microcontrolador, ya que únicamente se limitan a enviar unas cuantas microinstrucciones y este ejecuta lo necesario para llevar a cabo la acción. En caso de encender o apagar una válvula, se hace mediante la microinstrucción "output_low(pin)"para apagar y "output_high(pin)"para encender, las cuales reciben un pin, el cual se conoce gracias al diagrama de conexiones de la tarjeta. El manual del compilador especifica que se debe utilizar el nombre del pin a controlar, que en este caso es el PIN_A2 para la válvula A y PIN_A3 para la válvula B.

Para la utilización del DAC (figura 3.6), el proceso es un poco más complejo que el anterior, aunque en cierto punto se utilizan para algunas cuestiones las mismas microinstrucciones “output_low(pin)”y “output_high(pin)”.

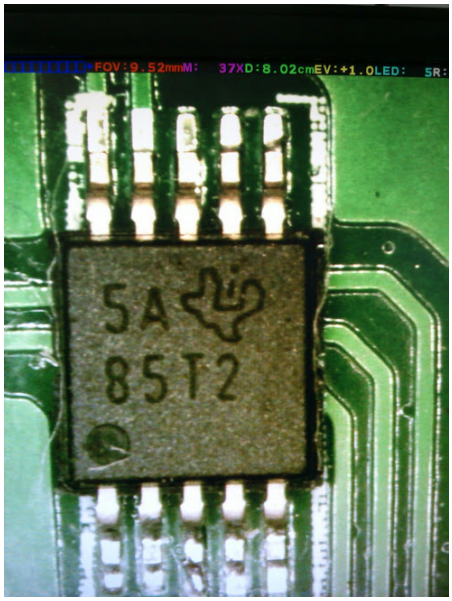

Figura 3.6: Detalle del DAC bajo el microscopio

Para poder hacer uso del DAC, se debe llevar a cabo la siguiente secuencia para tener el comportamiento deseado, el PIN_B0 del microcontrolador se encuentra conectado a la sección de SYNC, al ponerlo en estado alto o encendido, se prosigue a poner el PIN_B2 en bajo para que el DAC entre en modo de carga LDAC, inmediatamente el PIN_B1 encargado de poner en CLEAR el DAC. El PIN_B0 regresa a su estado original en bajo y se puede empezar a escribir valores (figura 3.7).

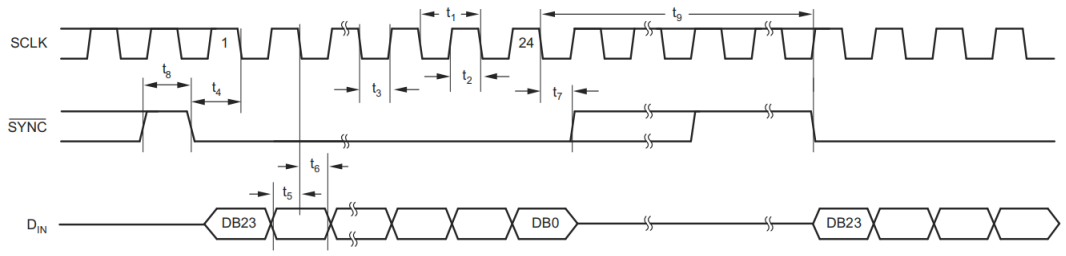

Figura 3.7: Se muestra la secuencia de escritura en el DAC, recuperado de [1]

Mediante el protocolo SPI se escriben (figura 3.8) los valores recibidos en el comando de control, previamente ya se había llevado a cabo un tratamiento del comando, esto con la finalidad de separar este, ya que como se muestra en la anterior los datos de escritura en el DAC se realizan "al revés", es decir el primer digito leído será el último en entrar al DAC.

Control Matrix

D23	D22	D21	D20	D19	D18	D17	D16	D15	D14	D13–D0	DESCRIPTION
Reserved	Reserved	Load B	Load A	Don't Care	Buffer Select	PD1	PD0	MSB	MSB-1	MSB-2... LSB	
(Always Write 0)					0 = A, 1 = B						
0	0	0	0	X	#	0	0	Data			WR Buffer # w/Data
0	0	0	0	X	#	See Table 3		X			WR Buffer # w/Power-down Command
0	0	0	1	X	#	0	0	Data			WR Buffer # w/Data and Load DAC A
0	0	0	1	X	0	See Table 3		X			WR Buffer A w/Power-Down Command and LOAD DAC A (DAC A Powered Down)
0	0	0	1	X	1	See Table 3		X			WR Buffer B w/Power-Down Command and LOAD DAC A
0	0	1	0	X	#	0	0	Data			WR Buffer # w/Data and Load DAC B
0	0	1	0	X	0	See Table 3		X			WR Buffer A w/Power-Down Command and LOAD DAC B
0	0	1	0	X	1	See Table 3		X			WR Buffer B w/Power-Down Command and LOAD DAC B (DAC B Powered Down)
0	0	1	1	X	#	0	0	Data			WR Buffer # w/Data and Load DACs A and B
0	0	1	1	X	0	See Table 3		X			WR Buffer A w/Power-Down Command and Load DACs A and B (DAC A Powered Down)
0	0	1	1	X	1	See Table 3		X			WR Buffer B w/Power-Down Command and Load DACs A and B (DAC B Powered Down)

Figura 3.8: Configuraciones para escribir valores en el DAC, tomado de [1]

En la tabla de la figura se muestra la disposición lógica del comando para ser ejecutado en cualquier parte del DAC, ya sea A o B.

Para una implementación sencilla, siempre se utiliza el modo de carga de datos independientes, es decir se trabaja con el DAC A o DAC B independientemente, a pesar de poder trabajar en alguna configuración más sofisticada.

Finalmente, el pin de sincronización se le asigna un valor alto y se evita que el DAC siga escuchando datos.

Con la finalidad de que el sistema tuviera un correcto funcionamiento y los datos enviados por el microcontrolador hacia el DAC fueran los esperados, se llevaron a cabo pruebas de escritura en el DAC, utilizando un osciloscopio digital, en el cual se visualizan las entradas y salidas de ambos dispositivos.

A continuación (figura 3.11), se muestran algunas capturas del sistema en funcionamiento, recibiendo, interpretando y ejecutando comandos relacionados a la utilización del DAC, gracias a su respectivo código de programado en el IDE (figura 3.9).

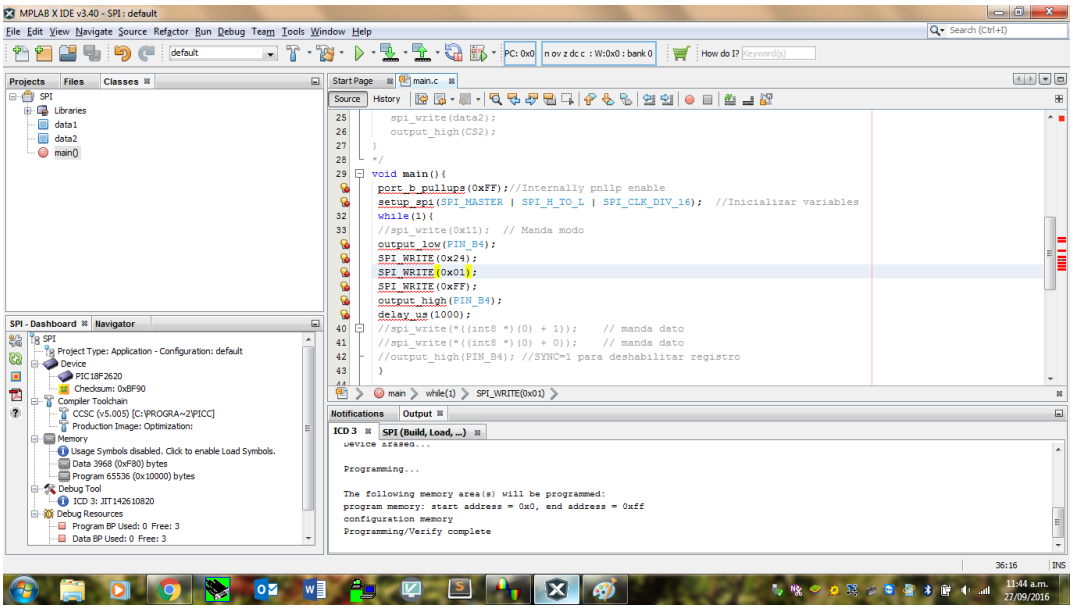

Figura 3.9: Captura del código de prueba en el IDE

```
void main(){
        port_b_pullups(0xFF);
        setup_spi(SPI_MASTER | SPIT_H_TO_L | SPI_CLK_DIV_16);
        while(1){
       SPI_WRITE(0x24);
       SPI_WRITE(0x01);
       SPI_WRITE(0xFF);
        output_high(PIN_B4);
        delay_us(1000);
        }
}
```

Código 3.7: Porción del código de ejemplo mostrado en el IDE (figura 3.9)

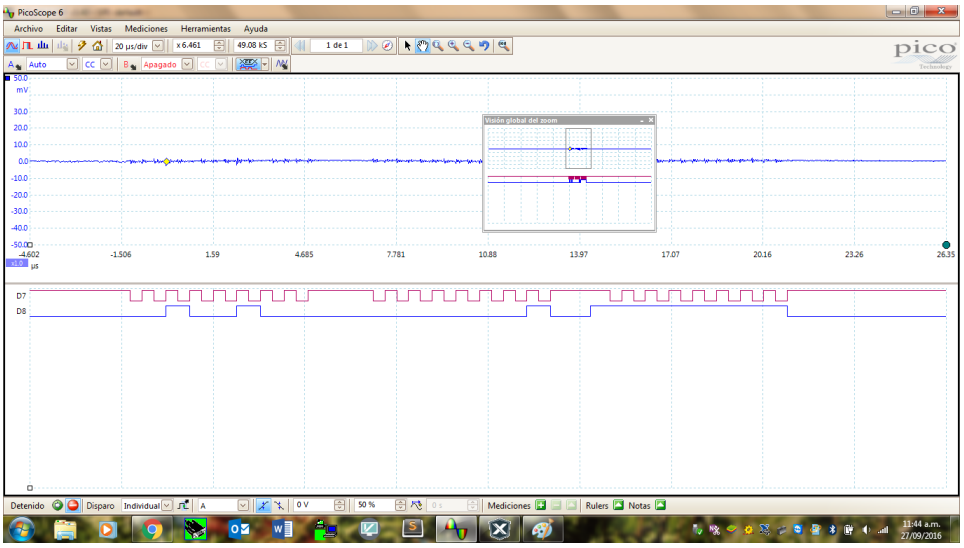

Figura 3.10: Muestra del comando ejecutado visto en un osciloscopio digital

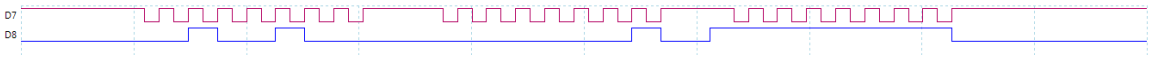

Figura 3.11: Detalle de la señal visualizada en la (figura 3.10)

En la figura 3.10 se muestra el comportamiento de los bits de control necesarios para la escritura en el DAC, D7 hace referencia al reloj del DAC. D8 es la señal generada por los valores 0x01 y 0xFF. Ambos vistos a detalle en la figura 3.10.

De esa manera se escriben todos y cada uno de los valores en el DAC, a continuación, se explican los valores correspondientes a la figura anterior.

Para llevar a cabo las pruebas de comandos sin tener finalizado el sistema, se utilizó una terminal (figura 3.12) para capturar, controlar y depurar los flujos de comunicación entre la tarjeta controladora y una computadora. La interfaz de la terminal es bastante compleja, sin embargo, se encuentra bien distribuida, permitiendo una navegación y comprensión rápida.

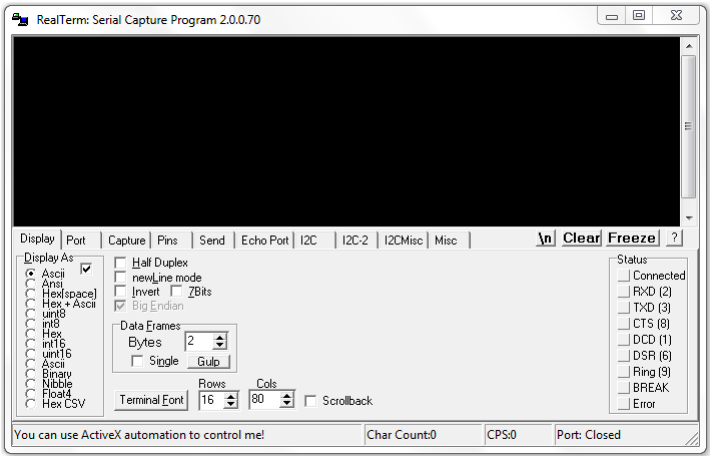

Figura 3.12: Captura del programa RealTerm para pruebas de envío de comandos

La parte en color negro es la terminal, y contraria a algunas otras terminales, en ella no es posible interactuar directamente, únicamente servirá para mostrar mensajes que sean escritos por la tarjeta controladora.

Para poder utilizar esta terminal, se requiere realizar lo siguiente:

- Abrir el programa de terminal Real Term.
- Configurar el puerto en donde se establecerá la conexión. (figura 3.13)

Llevando a cabo ese par de sencillos pasos, se puede proceder a enviar comandos.

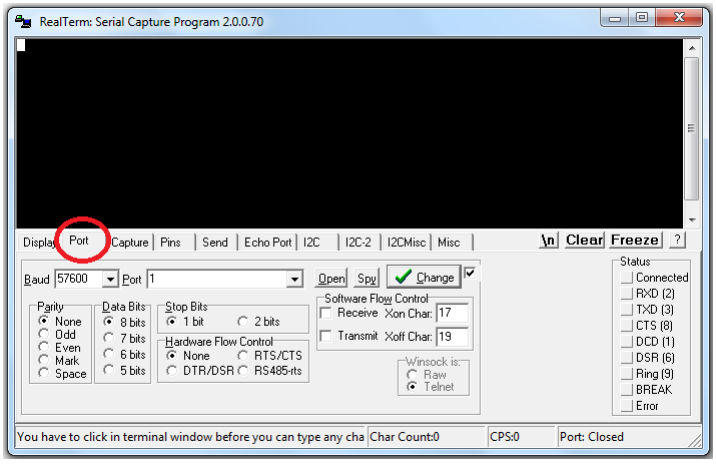

Figura 3.13: Muestra de selección de puerto

Se deberá seleccionar la pestaña encerrada en el círculo rojo "Port", en la sección Baud se utilizará 9600, lo cual es 9600 [Bd]. La velocidad en baudios es la velocidad a la que se transfiere la información en un canal de comunicación. "9600 baudios" significa que el puerto serie es capaz de transferir un máximo de 9600 [bit/s].

"Port" hace referencia a donde se encuentra conectado el adaptador utilizado para conectar la tarjeta con la computadora, normalmente se asocia a un puerto COM, el cual será seleccionado. Si existe más de un puerto COM, se deberá revisar a cuál puerto está asociado la conexión.

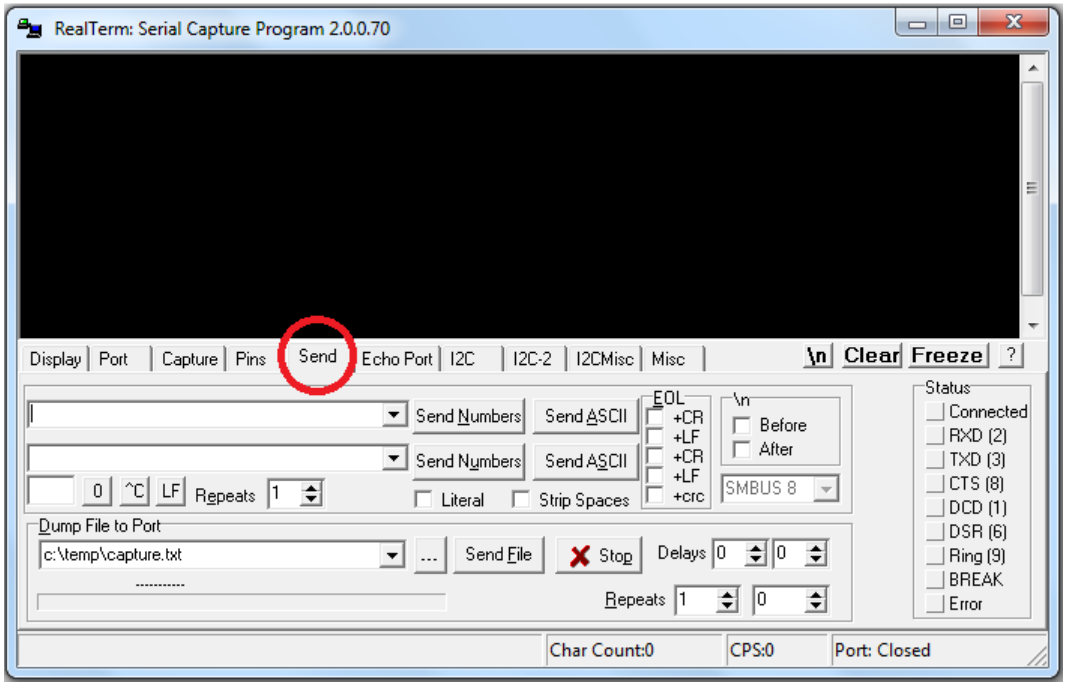

Figura 3.14: Selección del modo envió de comandos

Para proceder a enviar comandos, se selecciona la pestaña de "Send" (figura 3.14) y en el renglón blanco que aparece debajo del círculo rojo, se podrán enviar comandos de prueba, ya sea bien formados o comandos erróneos, todo con la finalidad de probar los distintos casos que se podrían presentar en la puesta en marcha del sistema.

Cabe mencionar que para utilizar en envió de comandos, se deberá de especificar los caracteres de control que se requieren para que la tarjeta interprete los comandos. En este caso se marcarán las casillas de "+CR" y "+LF", caracteres de los cuales se había hablado previamente al inicio de este capítulo.

Finalmente se escribe el comando en el recuadro seleccionado con el ovalo rojo (figura 3.15) y se da clic en el botón "Send ASCII".

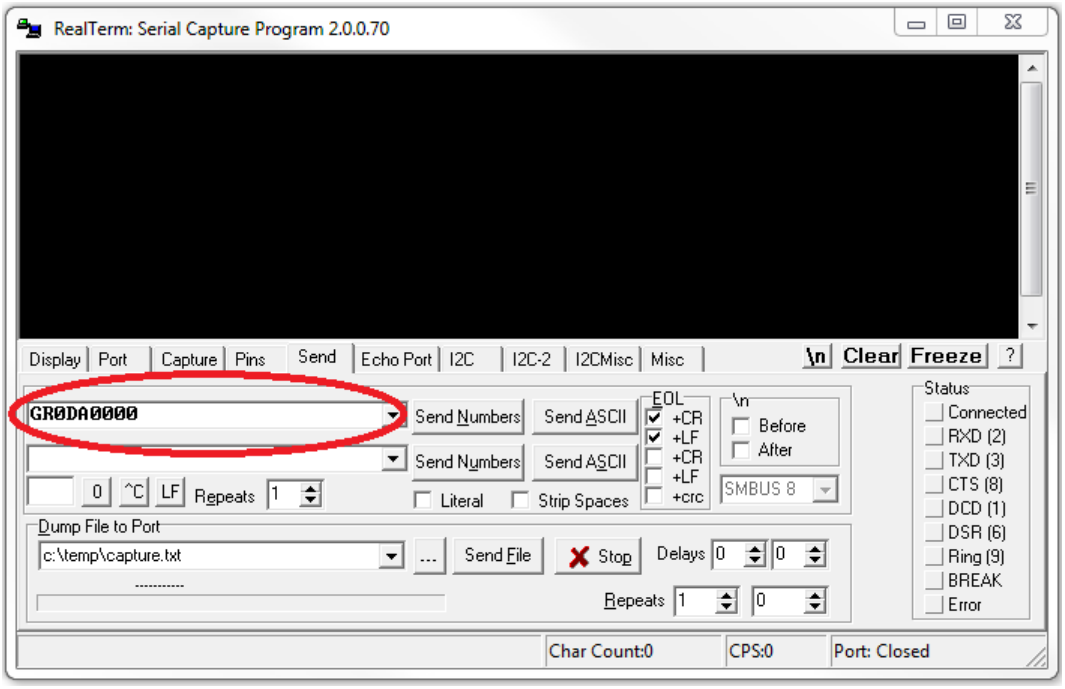

Figura 3.15: Sección de envió de comandos

Las pruebas llevadas a cabo con comandos relacionados con la utilización del control de las válvulas únicamente podían visualizarse al momento de enviar un comando válido en la terminal, para corroborar que se había ejecutado dicho comando y cómo se había interactuado con un relevador, el resultado esperado constaba de un clic emitido por el relevador asociado a cada válvula, ya sea al encenderla o apagarla. Para la utilización del DAC a pesar de contar con una señal caracterizada en el osciloscopio aun hacía falta corroborar que se tuviera un valor de presión asociado al valor escrito en el DAC, por ello se llevaron pruebas de presión contra voltaje de salida en los dos canales del DAC (figura 3.16).

Con la realización de las pruebas únicamente se pudo corroborar el voltaje que proporciona el DAC si realizaba un cambio de presiones, aunque en este caso la presión no es equivalente al voltaje suministrado, esto se debió a que la presión de alimentación no era suficiente para mantener el valor especificado por el DAC, así que únicamente se corroboró el funcionamiento de los reguladores de presión siendo

Figura 3.16: Multímetro contra manómetro

controlados por el DAC, además de corroborar que la tarjeta era capaz de recibir comandos asociados con el uso del DAC y su correcto funcionamiento.

3.4. Interacción entre software y firmware

Se presenta únicamente cuando existe intercomunicación (mediante dos o más dispositivos), en este caso entre una tarjeta controladora y una computadora (con al menos un puerto USB disponible). La cual será posible bajo ciertas condiciones, tales que ambos dispositivos se encuentren energizados, conectados mediante una interfaz especial y ejecutando un programa específico para comunicarse.

3.4.1. Protocolos de comunicación

Cada dispositivo cuenta con una interfaz de comunicación diferente, la computadora cuenta con un puerto USB como mínimo, mientras que la tarjeta controladora dispone de tres pines encargados de llevar a cabo una transmisión de datos mediante el protocolo RS485.

Para que ambos puedan comunicarse de manera transparente, se utiliza un convertidor RS485 a RS232, el cual se encarga de efectuar la comunicación a distancia entre ambos dispositivos. Dicho convertidor no requiere de una alimentación externa, además de que, para controlar varias tarjetas controladoras, solo se necesita de un convertidor, ya que las tarjetas pueden comunicarse entre ellas mediante el protocolo RS485.

A continuación, se muestra un diagrama de la topología que será utilizada para la conexión (figura 3.17).

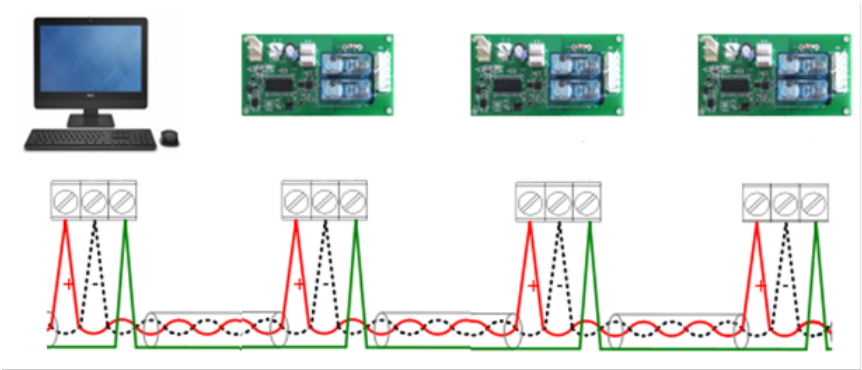

Figura 3.17: Topología para las tarjetas controladoras

Por tratarse de un protocolo estándar, la implementación y utilización de este, se hace de una manera generalizada. Nos permite tener conexión multipunto, tener una misma alimentación de 5[V], y dependiendo la longitud del medio donde se transmite, se tendrán diferentes velocidades de transferencia, por ejemplo, la velocidad máxima que se puede tener es de 10 [Mbit/s] a 12[m] o una longitud máxima de 1200[m] a 100 [kbits/s].

3.4.2. Interpretación de comandos

Este proceso se lleva en una computadora una vez que la tarjeta controladora envió datos por el medio de interconexión, lamentablemente al realizar únicamente la conexión física no existe una interacción en ambos sentidos. Para poder hacer uso de los datos recibidos mediante el puerto serial, se utiliza un software de instrumentación virtual llamado LabVIEW, en el que disponemos de diversos componentes (llamados instrumentos) los cuales sirven para diseñar la interfaz visual de los proyectos, así como los controles encargados de dotar de funcionalidad a lo diseñado.

LabVIEW permite más que únicamente interactuar con el puerto USB de la computadora, también provee de distintas herramientas y extensiones encargadas de proveer un marco de desarrollo bastante amplio. Al mismo tiempo es capaz de generar un archivo ejecutable, el cual será utilizado para que al programa final no se le consigan hacer modificaciones, asegurando que el usuario final únicamente tenga interacción con algo finalizado.

Para la interacción con el software de la tarjeta controladora se elaboraron diversos programas para probar la funcionalidad independiente al sistema final, con el fin de asegurar que la comunicación se diera de manera adecuada.

A continuación, se ilustra un programa de prueba, elaborado para realizar la interacción básica entre el microcontrolador y la computadora (figura 3.18).

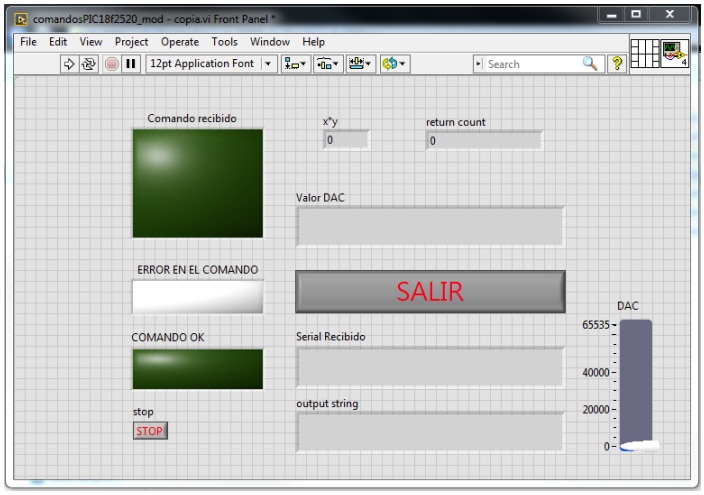

Figura 3.18: Vista el programa de prueba en LabVIEW

Lo que se muestra es únicamente la interfaz gráfica utilizada para las pruebas, en ella se encuentran algunos indicadores visuales en color verde o blanco, los cuales se "encienden" según sea el caso, además de contar con una sección de envió de escritura de datos en el DAC de la tarjeta controladora, los cuales van desde 0 a 65535 (número máximo que puede ser escrito en el DAC).

La interfaz no es nada compleja y la interacción es casi nula, sin embargo, permite verificar los puntos básicos de interconexión entre ambos dispositivos, listados a continuación:

- Corroborar que exista conexión entre los dispositivos, ya que al momento de hacer el respectivo cableado podría existir confusión entre los cables.
- Verificar la secuencia de datos que escribe la tarjeta controladora en la computadora, leer los datos recibidos en el puerto serial y además brindar el panorama actual de lo recibido.
- Llevar a cabo la escritura básica de comandos desde la computadora a la tarjeta controladora y poder realizar el seguimiento de las respuestas de esta.

Lo explicado anteriormente no puede llevarse a cabo si no se cuenta con algo encargado de controlar dicha interfaz gráfica, actualmente y como se ilustró, no cuenta con alguna funcionalidad, por lo que se mostrará cómo se comporta internamente dicha interfaz, lo cual pareciera que se trata de viñetas de una historieta, contando en cada cuadro la forma en que se va desarrollando la interacción, todo esto es posible gracias a que LabVIEW provee las herramientas necesarias para que la parte visual como la funcional interactúen de manera simple y directa.

Se llama diagrama de bloques a lo utilizado en LabVIEW para dotar de funcionalidad a la parte mostrada previamente.

En este caso, el diagrama de bloques es considerablemente más grande (al menos visualmente) que la interfaz gráfica, a continuación (figura 3.19), se muestra una captura de dicho diagrama, del cual se explicarán los componentes más característicos y necesarios para su utilización.

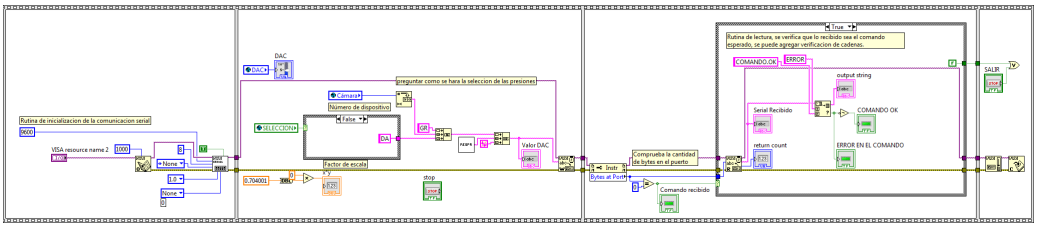

Figura 3.19: Secuencia del instrumento virtual

Se puede observar una secuencia, la cual es necesaria para que la tarjeta pueda enlazarse de manera correcta y consiga interactuar sin contratiempos, intencionalmente se capturó de esta forma la figura para hacer evidente el gran tamaño del elemento controlador, y todos los subcomponentes que lo hacen funcionar.

En el primer cuadro de la secuencia figura 3.20 (izquierda a derecha) se utiliza un objeto de LabVIEW llamado "VISA SERIAL", encargado de inicializar una sesión VISA, lo cual nos permitirá interactuar con diversos puertos que contenga la computadora, en este caso se usará el puerto USB.

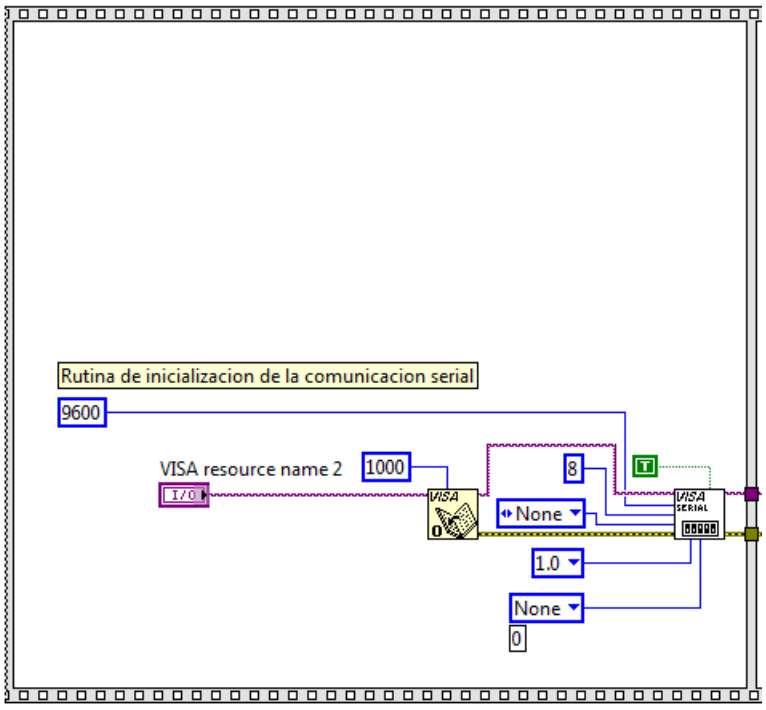

Figura 3.20: Secuencia inicial

Los pasos básicos para que la sesión VISA se desarrolle de manera correcta son los siguientes:

- Abrir una sesión con un puerto especifico.
- Configurar el puerto con campos como la velocidad de transferencia, caracteres de control y cantidad máxima de caracteres a leer.
- Llevar a cabo lecturas o escrituras en el dispositivo.
- Finalizar la sesión.
- Hacer el respectivo tratamiento de errores que pudieron presentarse.

En el siguiente cuadro (figura 3.21) y siempre que exista una conexión satisfactoria, ya se pueden enviar valores a la tarjeta controladora. Como se puede observar se presentan líneas del mismo color correspondientes al cuadro anterior, lo que hace referencia a que se siguen utilizando en esta parte del programa.

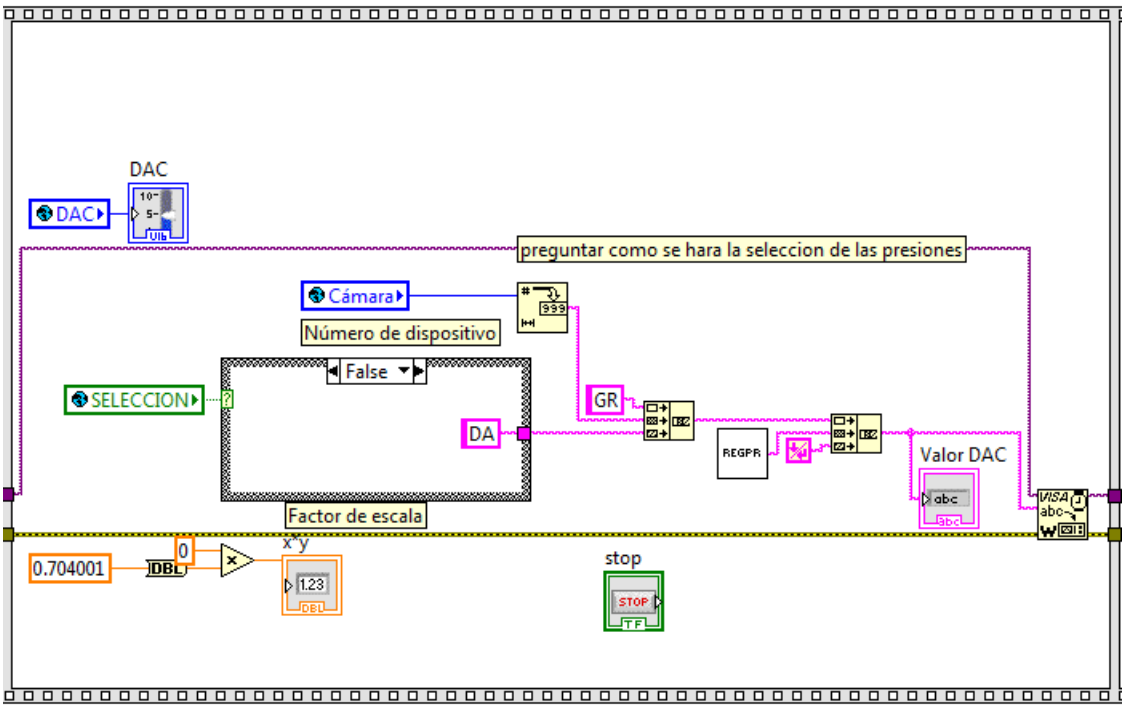

Figura 3.21: Segundo cuadro de la secuencia

Los cuadros pintados en color rosa contienen partes del comando a escribir en la tarjeta (cadenas de texto plano), y dependiendo su valor será la acción que realicen, en este caso se forma la primera parte de un comando "GR0DA" + los valores en hexadecimal que llegan desde el componente REGPR, un subcomponente encargado de dicha operación y que es un ejemplo de la aplicación final donde se integrará el programa.

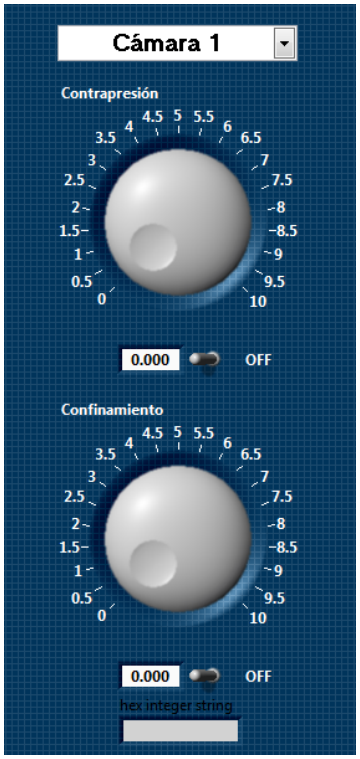

Figura 3.22: Perillas de control de presión

En dicho subprograma se lleva a cabo el tratamiento de datos expresados en [kg/cm^2] (figura 3.22) para tener una correspondencia directa en formato hexadecimal, el cual lee e interpreta el DAC para llevar a cabo sus respectivas tareas.

Una vez que el comando tenga la forma establecida en el subtema 5.3, se prosigue a escribirlo mediante un bloque VISA de escritura, el cual puede verse al final de la secuencia del cuadro descrito. Se utilizo este tipo de componentes simplificados con la finalidad de ahorrar espacio en los programas de pruebas además de ser una parte del programa final a donde se integrará el desarrollo final.

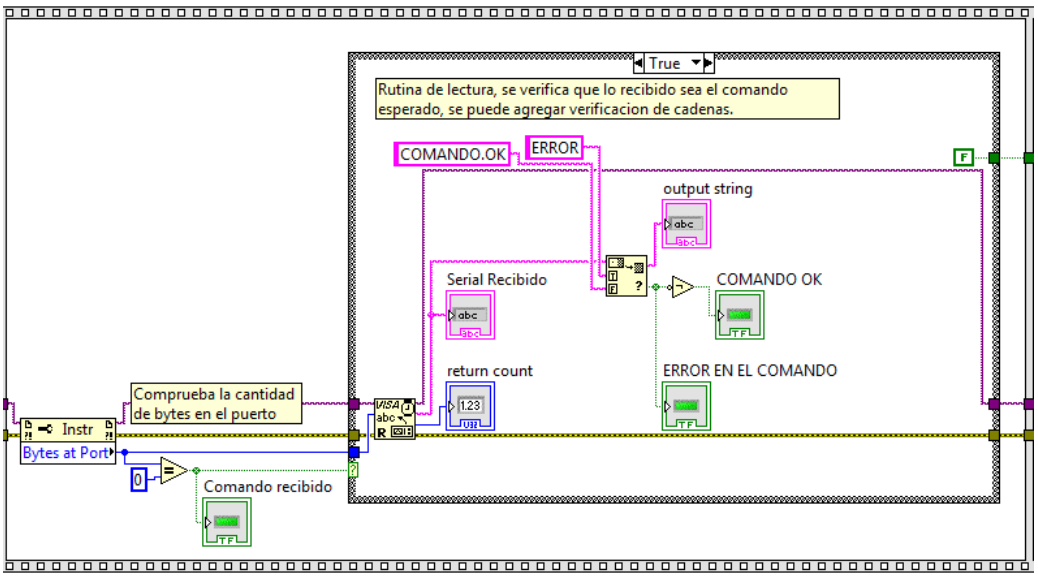

Figura 3.23: Tercer cuadro de la secuencia

Para el tercer cuadro (figura 3.23) se sigue el camino de los componentes VISA y se continúa haciendo el tratamiento de los datos, por tratase de un programa con fines de pruebas, no se fue estricto con el control de los comandos, ya que únicamente se quería verificar conexión e interacción básica. Dada esta funcionalidad se podía recibir y enviar comandos prácticamente al mismo tiempo, así que el control de envío y recepción de comandos lo brindaba el usuario al delimitar si enviaría o recibiría.

Finalmente, y sin importar la operación realizada, la sesión VISA se daba por terminada, esperando que al final se mostrará o no, dependiendo el caso, si existió algún error, esto se observa en la figura 3.24.

Las pruebas de conexión entre una computadora y una tarjeta controladora se fueron haciendo sin tener las conexiones finales en donde estaría albergado el sistema final, sin embargo, con la ayuda de LabVIEW y en ocasiones con un osciloscopio digital, se logró llevar a cabo gran parte de pruebas, las cuales dieron lugar a que la integración final se diera de una manera rápida y sin contratiempos destacables.

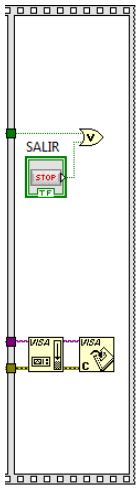

Figura 3.24: Cuadro final de la secuencia

3.4.3. Respuesta de comandos

La respuesta de comandos va directamente asociada con la sección anterior referente a la interpretación de comandos. Tanto en la tarjeta como en LabVIEW, es posible proveer códigos de error, es decir, que uno puede leer el error de otro y viceversa.

La primera versión del firmware contemplaba tener los siguientes estados, capaces de dar un panorama tras la ejecución de algún comando.

- "OK #D"
- "OFF Valv_A #D"
- "ON Valv_A #D"
- "OFF Valv_B #D"
- "ON Valv_B #D"
- "W_DAC_A #D"
- "W_DAC_B #D"
- "ERR #D"

Cada código contiene la información mínima necesaria para identificar el estado del comando, tales que la primera sección indica el tipo de evento al cual se encuentra asociado dicho código. Inmediatamente (con excepción del primer y último código), se indica referente a que está asociado, ya sea válvula o DAC. Finalmente se indica desde donde se suscitó dicho código, es decir el número de dispositivo.

Además se optó por reducir los estados a únicamente "OK" o "ERR" con su respectivo número de dispositivo, ya que buscaba no saturar el medio por donde viajan los datos y perder comandos al trata de leer códigos de ejecución.

```
void msjEstado(int estado)
{
        output_high(PIN_C1);
        delay_ms(10);
        printf("D%c%s\r",addressDisp,cmdEstados[estado]);
        output_low(PIN_C1);
}
```

Código 3.8: Subrutina para imprimir estado de ejecución de comando

Lo que se realiza en la subrutina, es detener la trasmisión de datos y quedar en modo recepción, se esperan 10 [ms] para evitar que queden datos pendientes de transmitir del comando anterior y se prosigue a "imprimir" el mensaje que contiene la dirección del dispositivo que ejecutó ese comando y el estado que se toma del siguiente arreglo de estados.

```
static char * cmdEstados[] =
{
        "OK",
        "off_valvA",
        "on_valvA",
        "off_valvB",
        "on_valvB",
        "W_DAC_A",
        "W_DAC_B",
        "ERR"
};
```

Código 3.9: Arreglo de estados

Siempre que se envié un comando con las reglas establecidas y los parámetros necesarios para ser ejecutado, se ejecuta también la subrutina con el mensaje de estado 0, el cual corresponde a “OK” en caso contrario se enviará el mensaje de estado 7, el cual hace referencia a “ERR”, en ambas ejecuciones se especifica en que número de dispositivo se presentó dicho estado.

Finalmente, tras haber enviado el mensaje de estado, se prosigue a regresar a la tarjeta a modo transmisión, para continuar con su ejecución habitual.

Capítulo IV

Interfaz de usuario (software)

Una vez explicado cómo funciona el firmware, cómo fue programado el microcontrolador, las tareas que lleva a cabo y las pruebas realizadas para verificar su correcto funcionamiento, se procederá a explicar el programa final al que fue integrado, las tareas que realiza y como interactúa el usuario final con él.

Además, se mostrará el software a donde se adaptó el desarrollo elaborado en esta tesis, así como el proceso de su utilización y su importancia para las actividades que lleva a cabo la gente del laboratorio de mecánica de suelos.

4.1. Software controlador

La tarjeta controladora posee un firmware capaz de leer e interpretar comandos, para adaptar el desarrollo a lo que se encontraba en el laboratorio donde se instalaría, también se solicitó elaborar un software encargado de escribir los comandos necesarios a la tarjeta para que esta se encargara de interpretarlos y ejecutarlos.

Por la naturaleza del sistema con el que se contaba donde sería integrada, no se podía permitir que los comandos fueran ingresados manualmente, es decir mediante una terminal, tal como se hizo en las primeras pruebas, o que interpretará comandos únicos por cada ejecución de programa, como se mostró en la interacción de software y firmware. Fue por ello por lo que se elaboró un programa (figura 4.1) que pudiera integrarse al software actual. El cual sería capaz de tomar parte de todos los requerimientos y adaptarse de una manera transparente para el software actual, sin hacer

mayores modificaciones o intervenir con lo que ya se tenía hecho.

El programa elaborado para la comunicación posee al igual que el programa de ejemplo, un módulo de comunicación NIVISA, el cual crea una sesión para dicho dispositivo. El programa solo es ejecutado cuando se le requiere, aunque siempre se mantiene a la escucha para recibir todo lo que se envié, sin embargo, es capaz de descartar los comandos que no le corresponden, provocando que cuando sea un comando válido, lo ejecutara inmediatamente.

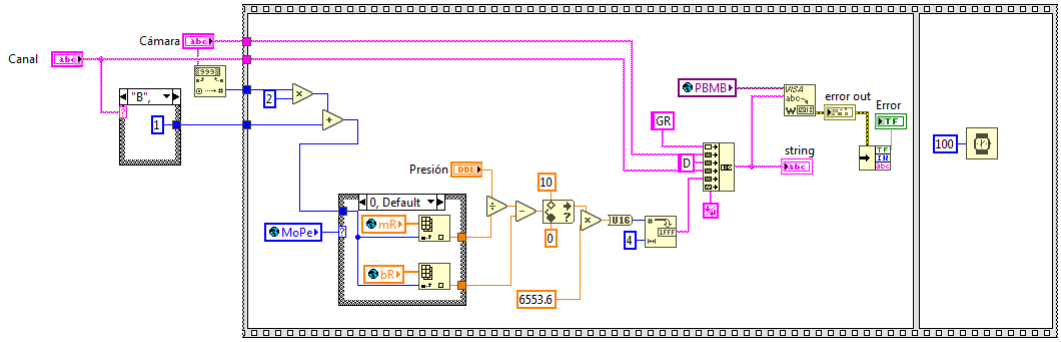

Figura 4.1: Diagrama del software que controla el DAC de la tarjeta

En dicho software podemos ver el diagrama de bloques correspondiente a todas las funciones que desempeña. Existe una comunicación serial, la cual se encarga de transmitir los comandos hacia la tarjeta y viceversa, también podemos ver que dicha comunicación se encuentra presente en todo momento, esto con la finalidad de poder tener un seguimiento del comando enviado y poder captar errores en la línea de comunicación.

A su vez, podemos ver pequeños cuadros que interactúan con el sistema, estos son otros instrumentos virtuales que se encargan de llevar a cabo otra tarea. Pero por cuestiones de simplificación, se reduce toda la lógica a un solo elemento. Existen por otro lado, demás elementos que tienen una lógica muy reducida, por ejemplo, los apagadores de las válvulas, los cuales únicamente tienen esa función, la de apagar o encender las válvulas. Tal cual como se realizaría el proceso de manera física, los apagadores pueden tener dos valores, encendido o apagado. La lógica para que los apagadores funcionen a la par de la tarjeta, es asociando la parte del comando que les correspondería ejecutar, ya que cada apagador se encarga de interactuar con una válvula, en este caso, se asociará con la cadena del comando equivalente a “V0” o

"V1", lo cual será una constante siempre, la única variable de esta acción corresponde al valor de encendido o apagado de cada válvula, lo cual complementaria el comando tal como se explicó en capítulos anteriores. Por lo tanto, los apagadores proveerán dicha constante, siendo un "1" cuando estén encendidas o un "0" cuando se encuentren apagadas. Es decir, se asociarán esos estados a una cadena de caracteres, la cual contendrá finalmente un comando listo para ser enviado a la tarjeta controladora.

El programa descrito se encuentra en constante escucha de los posibles comandos que podrá ejecutar, es por ello por lo que se su ciclo de vida está determinado por la cantidad de comandos que tenga que ejecutar.

Este módulo también es de suma importancia para el desarrollo, dado que implica un menor procesamiento por parte de la tarjeta controladora, la cual solo se encarga de comunicarse y transmitir comandos, dejando las labores con más procesamiento a la computadora, ambas trabajan en sinergia y sin duda las operaciones que realiza el firmware no serían del todo posible sin el software, lo mismo aplica en forma inversa, el software no sería capaz de llevar a cabo en su totalidad las tareas para las que fue concebido, por lo tanto software como firmware son de vital importancia para este desarrollo.

Ahora se explicarán los componentes internos que controlan la tarjeta controladora, como es la interacción entre este controlador y las tarjetas, así como la composición de cada parte del software. Además de que se mostrará la parte visual y el flujo de datos desde la interfaz de usuario hacia el control lógico y finalmente el envió hacia la tarjeta controladora.

Dado que existen cuatro elementos distintos y cada uno se controla de manera independiente, se tiene que crear un caso de interacción para cada uno, así que LabVIEW proporciona un elemento para llevar esto a cabo.

Figura 4.2: Detalle de la interfaz que interactúa la tarjeta controladora

La interfaz mostrada en la figura 4.2 controla los componentes de la tarjeta de la siguiente manera:

- Perilla de confinamiento controla al DAC A
- El botón válvula en la sección de confinamiento controla la válvula V0 de la tarjeta
- Perilla de contrapresión controla al DAC B
- El botón válvula en la sección de contrapresión controla la válvula V1 de la tarjeta
- El rectángulo blanco indica el valor que escribirá la perilla en la tarjeta, se puede ingresar manualmente de forma directa el valor para un valor en específico

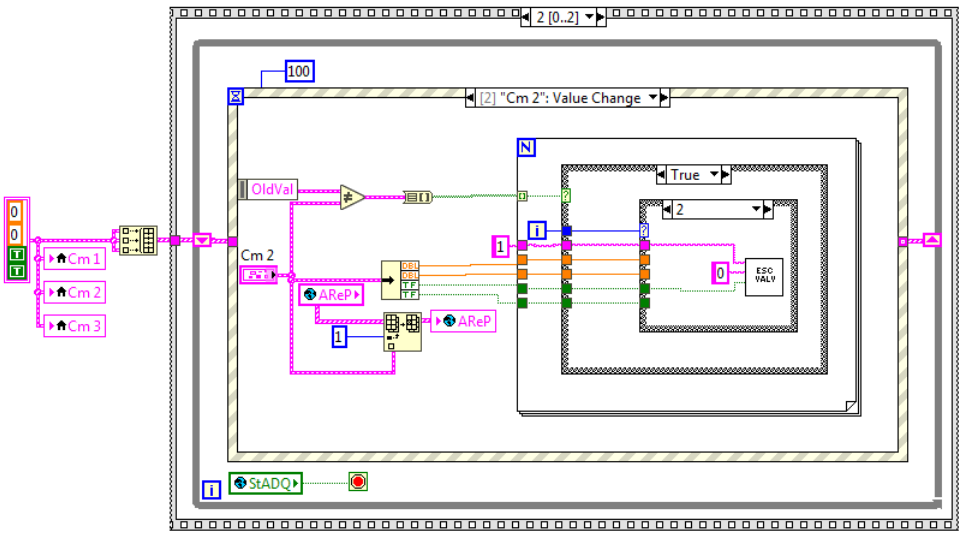

Figura 4.3: Bloque principal para selección de cámara y elemento a controlar

Debido a que el programa fue pensado para interactuar con tres tarjetas para controlar sus respectivas cámaras triaxiales, como lo muestra la figura 4.3, se tiene un primer filtro por selección de cámara, en este caso se trata del rectángulo más grande y que contiene los demás rectángulos de menor tamaño.

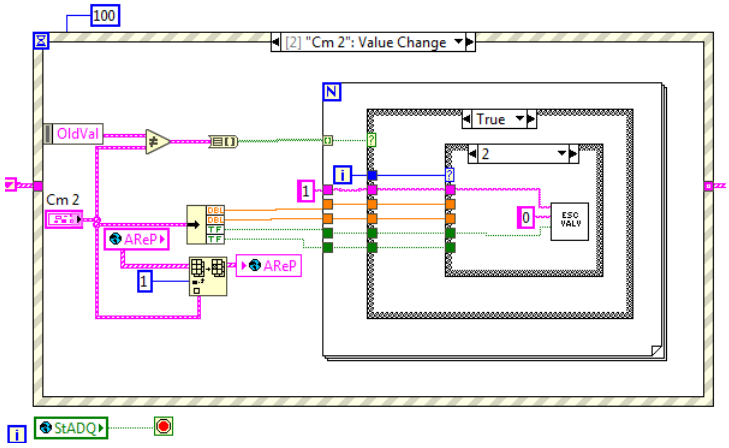

Figura 4.4: Detalle de la cámara dos

La cámara que se controlará con este software es la número dos, así que es la que se seleccionará y se explicará.

En el bloque de la figura 4.4 se observan diversos elementos, como algunos de la parte inferior con la etiqueta “StADQ”, el cual sirve para detener la transmisión de datos hacia la tarjeta, también a la salida del elemento “Cm 2” se observa que dicho elemento llega a un nuevo bloque con cuatro elementos, aquí es donde se lee el valor que se envió en el panel de la (figura 4.2) y dependiendo de donde llegue, será el valor escrito para continuar con el flujo de datos, aunque salen cuatro valores de este bloque, solo uno se enviará dependiendo de la acción deseada. Los cuatro valores mencionados se tratan de “DBL”, “DBL”, “TF” y “TF” corresponden al tipo de dato de “Confinamiento”, “Contrapresión”, “Válvula” y “Válvula”. Mientras que “OldVal” se utiliza para verifica si existió algún cambio en la variable “Cm 2” . También se encuentra dentro del recuadro un número “1” dentro de un recuadro rosa, esto hace referencia a que se trata de una cadena de texto y es la encargada de determinar que número de dispositivo se controlará.

En la parte más interna de la figura 4.3 podemos encontrar la parte encargada de controlar cada caso, ya sea una válvula o un DAC.

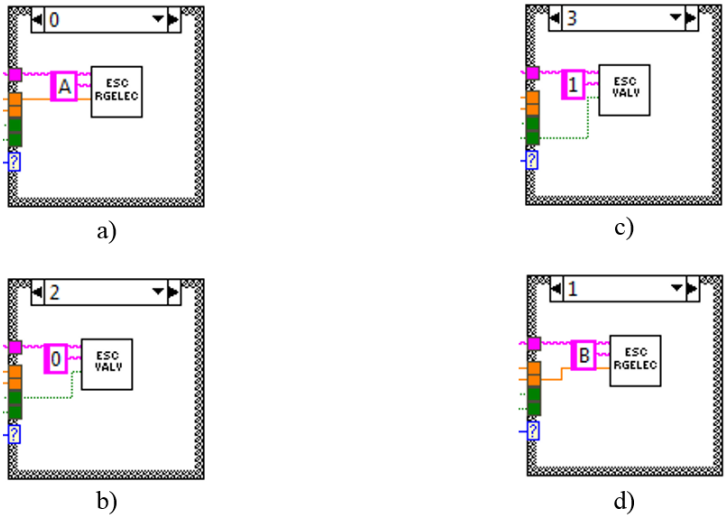

Figura 4.5: a) Selección DAC A b) Selección válvula 0 c) Selección válvula 1 d) Selección DAC B

Dependiendo de qué caso se haya seleccionado será el valor que se enviará a su respectivo módulo de interpretación de dicho valor, como se mencionó arriba, aunque en cada uno de los recuadros de la figura 4.5 convergen cuatro terminales, únicamente y dependiendo del caso, solo una determinara el valor que llegará a la última sección.

Existen las cuatro posibilidades anteriores, y aunque pareciera que hemos llegado al final, aún falta explicar el componente que se encuentra justo en medio de cada caso. Y a pesar de parecer un bloque único y pequeño, dentro de él se encuentra toda la lógica para controlar las válvulas o ambos DAC de la tarjeta.

Dado que se controlarán dos elementos de dos tipos distintos, se puede ver que los casos tienen en común un cuadro de "ESC VALV" y otro de "ESC RGELEC. El primero hace referencia al control de válvulas y el siguiente al manejo de los DAC de la tarjeta, que a su vez controlaran los reguladores de presión o servo válvulas.

En el panel controlador de válvula de la figura 4.6, se observan diversos componentes, y aunque el usuario no podrá interactuar directamente con ellos, se encargan de interpretar los valores que reciba de los elementos que lo controlan.

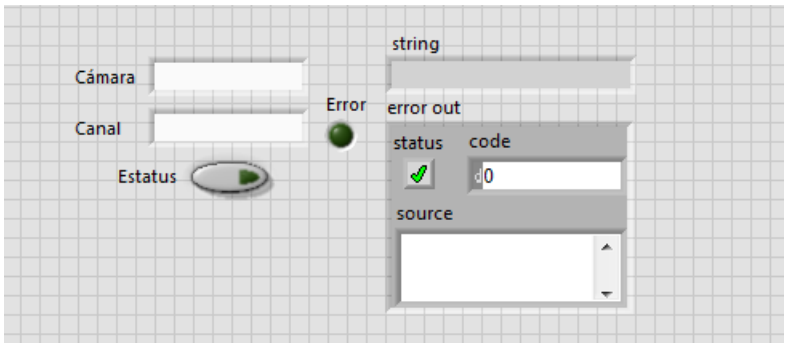

Figura 4.6: Panel controlador de válvulas

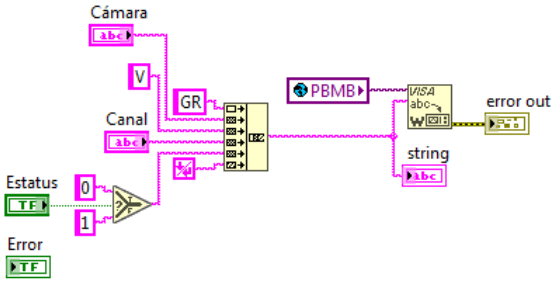

Figura 4.7: Diagrama de bloques que controla el panel de la figura 4.6

El tratamiento de determinar que válvula es la que se opera, se recibe del componente que contiene este elemento, en este caso el de la figura 4.5 b) o figura 4.5 c), valor que se toma como "Canal" referente a cuál válvula se operará. La variable "Cámara" es el número "1" de la figura 4.4, asociado al identificador de dispositivo a utilizar. Por último se lee el valor de "Estatus" (figura 4.7), que llega desde el botón del panel principal y dependiendo si se lee un valor verdadero o falso, será la cadena ("1" o "0") que se envié al bloque central donde se concatenan los caracteres y que finalmente formarán el comando que se escribirá en el bloque VISA.

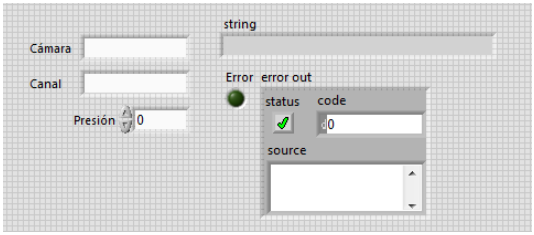

Figura 4.8: Panel controlador de los reguladores de presión

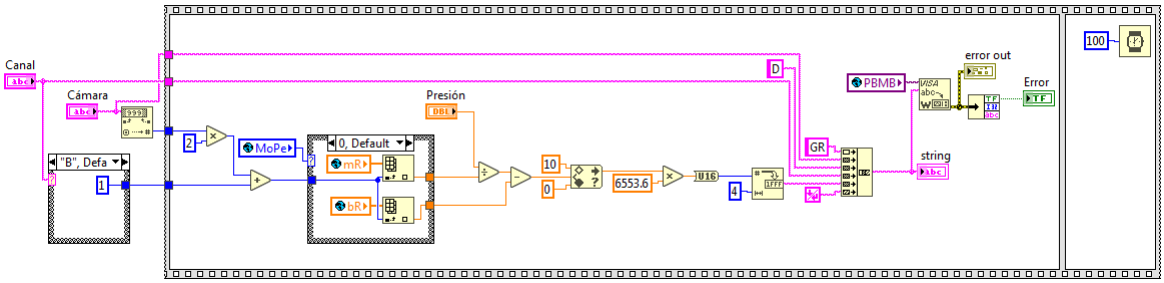

Figura 4.9: Diagrama de bloques del control de los reguladores de presión

Al igual que el controlador anterior, no se presenta interacción con el usuario en el panel (figura 4.8), y la lógica de funcionamiento se encuentra el diagrama de bloques que se presentará. En el control de válvulas, se retoman dos valores que se enviaron en la figura 4.5 a) y figura 4.5 d), de la misma manera se utiliza el valor de presión que se selecciona en alguna de las perillas de la figura 4.2.

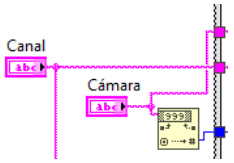

Figura 4.10: Detalle de la figura 4.9

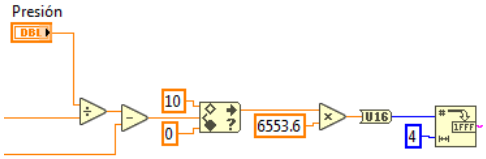

Figura 4.11: Tratamiento de la entrada recibida

Ya en esta parte del diagrama el programa se encarga de tratar los datos recibidos (figura 4.11) por la perilla de incremento, se realiza una comprobación de que el número recibido se encuentre dentro de un rango entre 0 y 10, ya que es el valor máximo que se puede escribir en la tarjeta controladora. Finalmente se multiplica el valor por un factor de escala, se convierte a un número de 16 bits y por último se convierte a un número hexadecimal de cuatro dígitos, los cuales serán enviados a la tarjeta controladora.

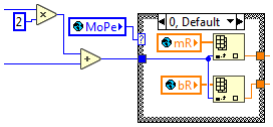

Figura 4.12: Parte final de la escritura en los reguladores de presión

Esta sección (figura 4.12) es la encargada de rescatar los respectivos valores de las constantes para calcular la presión, es decir las variables globales del sistema.

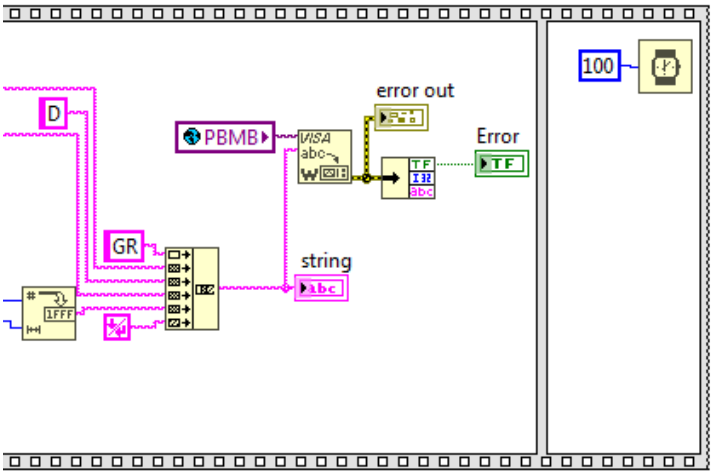

Figura 4.13: Parte final de la escritura en los reguladores de presión

Como se ilustra en la figura 4.13, existe un bloque encargado de concatenar los caracteres generados a lo largo de todo el flujo del programa, cumpliendo con lo establecido en el capítulo 5.3, donde se especifica como deben estar conformados los comandos.

Tras tener un comando válido y correcto, se prosigue a escribirse en la sesión VISA, la cual se encuentra previamente configurada por el programa principal que contiene estos submódulos que se presentaron.

4.2. Guía básica de uso e interacción con el usuario

Para utilizar dicho software y aprovechar las funcionalidades que la tarjeta puede desempeñar, es necesario explicar cómo funciona este pequeño modulo y la forma en que interactúa con el usuario.

El usuario final interactuará con la tarjeta mediante un panel como el mostrado en la siguiente figura 4.14:

Figura 4.14: Panel de configuración de la tarjeta controladora

En dicho panel se puede observar que se cuenta con dos perillas y dos interruptores por cada cámara a la que están asociadas. Sin embargo, únicamente se actualizaría

la cámara número dos, aunque a futuro se planea actualizar todas las cámaras. Independientemente de eso, cada uno de dichos elementos es capaz de controlar un periférico de la tarjeta controladora que tengan asociada. También se encuentra un rectángulo de color blanco, el cual sirve para ingresar la presión asociada a dicho dispositivo, sin embargo, al tratarse de una entrada de texto, se puede escribir el valor, prescindiendo de la perilla, todo con la finalidad de poder obtener valores con mayor precisión, ya que el movimiento de la perilla en momentos no permite ajustar el valor con exactitud.

La primera perilla es capaz de variar la presión que la servo válvula suministrará mediante la función de regular la cantidad de aire que proporciona, dichos valores están expresados en kg. El interruptor asociado a dicha perilla es capaz de dejar o no pasar dicho aire. Se comporta como una llave de paso, pero controlada electrónicamente mediante la tarjeta. Dicho botón controla una válvula de apertura y cierre, la cual como ya se vio, está relacionada directamente con un relevador de la tarjeta.

La segunda parte de la interfaz gráfica tiene características similares y es capaz de desempeñar las mismas tareas, con la diferencia de encontrarse asociadas a otra servo válvula y a otro interruptor. Esto se desarrolló de tal manera puesto que en el laboratorio de mecánica de suelos se buscaba realizar nuevos tipos de prueba, los cuales no serían posibles sin tener desarrollado este nuevo sistema.

Tal como una perilla común y corriente, dicha perilla (figura 4.15) es capaz de ser manipulada mediante el movimiento de algún periférico de entrada de la computadora, un ratón, por ejemplo; al igual que el interruptor (figura 4.16), con la facilidad de posicionar el cursor sobre él y con un simple clic, modificar su estado.

Figura 4.15: Detalle de la perilla con un valor seleccionado

Figura 4.16: Ejemplo de los estados que puede tener el interruptor

4.3. Interacción con el usuario

La interacción con el usuario con este desarrollo se presenta en muy pocas ocasiones, sin embargo, permite a los operadores de la cámara, iniciar las pruebas de una manera sencilla y desde la comodidad de un escritorio, ya que la actualización realizada en este trabajo, colaboró a que, en un par de minutos, se pudiera configurar una prueba, la cual, puede durar hasta quince días y únicamente requerirá de supervisión en contadas ocasiones.

El panel de configuración mostrado en la figura 4.15, reemplazó a elementos físicos que se tienen instalados en el laboratorio, se trata de manómetros de caratula y llaves de paso, las cuales se tenían que operar de manera manual y buscando una precisión meticulosa.

Figura 4.17: Panel de configuración para las cámaras sin actualizar

En la figura 4.17 se muestra el panel, que sigue realizando lo propio para las demás cámaras sin actualizar. Comparado con el panel en software de mostrado más arriba, la diferencia es ahorro de espacio es bastante, ya que únicamente se requiere de un monitor para poder ver los mismos datos que en este panel de configuración.

Una de las grandes ventajas de la actualización del sistema, es que, como ya se menciono anteriormente, se pueden automatizar las pruebas, y esto, aunque no es una de las funcionalidades del programa realizado, se explicará como es que ayudó en gran medida a realizar dicho propósito.

A continuación (figura 4.18), se mostrará el flujo de las pruebas realizadas, con el fin de ilustrar en donde se agregó el desarrollo realizado.

Figura 4.18: Interfaz del programa con nuevos cambios

Cabe mencionar que la interfaz de la derecha, los dos rectángulos horizontales, representan el programa al que se anexo el área mostrada previamente, es decir la parte de la izquierda, encargada de la regulación de la presión y la apertura y cierre de válvulas. Se pude apreciar que el diseño de la nueva interfaz va acorde a lo que ya se tenía, por lo tanto, también en el aspecto visual se trabajó para que no pareciera que se agregó un elemento ajeno, más bien para que diera la impresión de que siempre estuvo ahí.

Ya que se agregó este nuevo módulo de funcionamiento a la interfaz, también se añadieron nuevas posibilidades al sistema, ya que el firmware y software elaborados, dieron parte a una serie de pruebas que no era posible realizarlas, no por falta

de capacidad o conocimiento por parte de los encargados, más bien la limitante tecnología y las demandas de las pruebas lo hacían algo casi imposible; el primer punto que se requería para hacer la nueva prueba sin utilizar este sistema, es que se encontrara más de una persona encargada de la misma, ya que se requería variar dos presiones en tiempos similares, además de esto era necesario manejar dicho cambio de presiones cada cierto tiempo, aunado a esto se tenía que recabar datos a cada cambio de presión, para finalmente hacer los cálculos pertinentes.

Con la ayuda de esta actualización, todo consiguió hacerse de manera automática, la presión se puede variar automáticamente cada lapso y los valores obtenidos por diversos sensores se almacenan automáticamente en otra parte del programa de LabVIEW. Gracias a esto, los operadores de la cámara configuran la prueba dependiendo lo que se busca obtener y la tarjeta controladora trabajando en conjunto con el software hacen capaz llevar a cabo esta nueva prueba.

Y ahora, es posible realizar esta prueba, que transcurran cinco, diez o hasta quince días y los operadores de la cámara solo deberán supervisar la muestra que contiene la cámara y dar visto bueno de que el proceso se está realizando de manera satisfactoria.

Capítulo V

Integración y pruebas con el sistema final

Finalizadas las pruebas de funcionamiento del firmware y el software encargado de la comunicación con la tarjeta, se prosiguió a realizar la integración con los diversos componentes que contaba el sistema. Dichas pruebas consistieron en la interacción básica entre una computadora y la tarjeta controladora. La interacción se daba cuando la computadora enviaba un comando, la tarjeta lo interpretaba, y si era válido lo ejecutaba, finalmente la tarjeta respondía a la computadora un estado de "OK" o "ERROR", dependiendo cual fuera el caso.

5.1. Ajustes necesarios al sistema a actualizar

Una vez montadas y conectadas las servoválvulas y electroválvulas por parte del personal técnico encargado del laboratorio, se prosiguió a llevar a cabo la modificación del software que se tenía, para integrar la nueva fase de las pruebas.
En la figura 5.1 se mostrarán los cambios realizados.

Figura 5.1: Antes y después de actualizar la cámara triaxial

5.2. Integración con el sistema eléctrico y electrónico

La integración con el sistema eléctrico y electrónico se desarrolló de manera no invasiva, prácticamente el sistema se añadió sin nada destacable. Lo más complejo de este proceso fue elaborar los conectores para los periféricos a controlar, identificar qué y a donde debería conectarse.

Para los controladores de presión se añadieron conectores de 4 terminales a cada uno, relacionando cada terminal con su respectivo borne de la tarjeta controladora. Para la alimentación de la tarjeta únicamente se añadió un conector de dos terminales (figura 5.2) y se energizó desde una fuente de 24 [V].

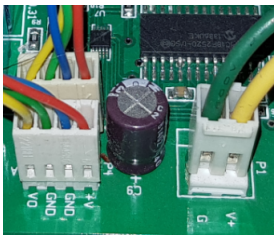

Figura 5.2: Detalle conector de la servo válvula (izquierda) y alimentación (derecha)

Para la integración eléctrica del sistema, únicamente se buscó un conector cercano de 127 [V] (figura 5.3), ya que en las pruebas no se contaba con un conector a la distancia requerida.

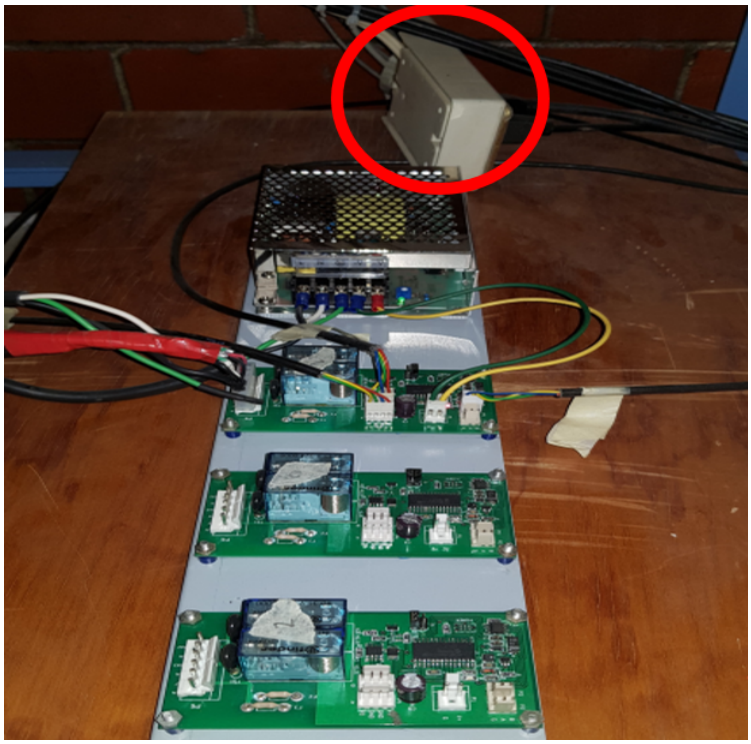

Figura 5.3: Detalle de la conexión a una toma eléctrica

5.3. Integración con el software

La integración de software será considerada como la adición de nuevas funcionalidades al programa con el que se contaba. Dado que ya fue explicado el software, tanto de pruebas como la versión final, ahora se abordará la forma en la que dicho programa se integró.

Gracias a los requerimientos dados por la coordinación de electrónica, la integración no debería presentar ningún problema, dado que se utilizó la plataforma sobre la cual ya operaba gran parte del sistema actual, así que, bajo esa condición, la integración sería transparente para el sistema actual.

Además, como se proporcionó un programa de muestra de un submódulo, lo cual proporcionó un marco de desarrollo muy acotado, prácticamente lo único que se debía considerar era el tipo de entradas y salidas, así como los argumentos que se le proporcionaban al sistema y resultado que entregaría.

Una vez desarrollado el software bajo los lineamientos mencionados, únicamente se proporcionó el código al coordinador de electrónica del instituto de ingeniería, encargado de intégralo al sistema del laboratorio de mecánica de suelos.

La sección del "código" que contiene al programa donde se integró fue la siguiente (figura 5.4):

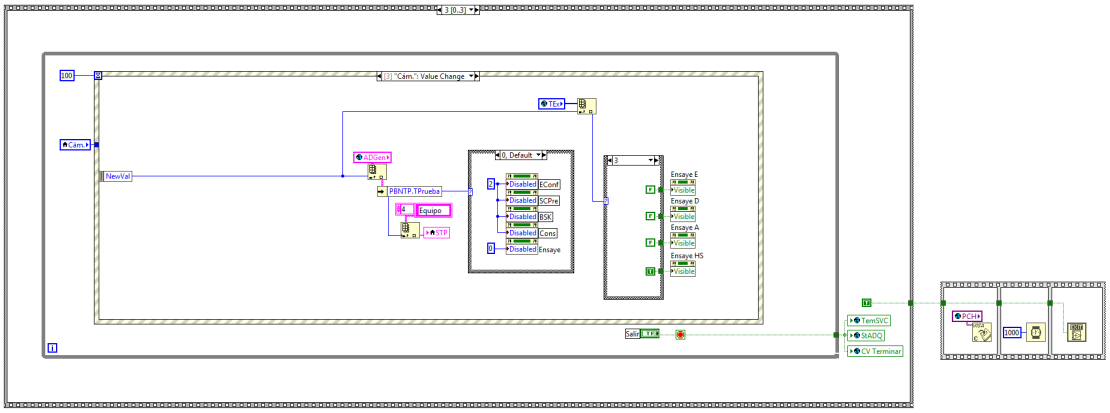

Figura 5.4: Sección donde se integró el código de LabVIEW

Cabe mencionar que en la figura anterior únicamente se muestra una mínima porción del programa al que se integró el desarrollo de este trabajo. Intencionalmente se

colocó la figura con esa proporción para que se distinguiera la cantidad de elementos que controlan al software actual, ya que se busca que se aprecie la magnitud del proyecto al que se le agregaron nuevas características para nuevas pruebas.

La parte que a donde se integró fue la siguiente (figura 5.5):

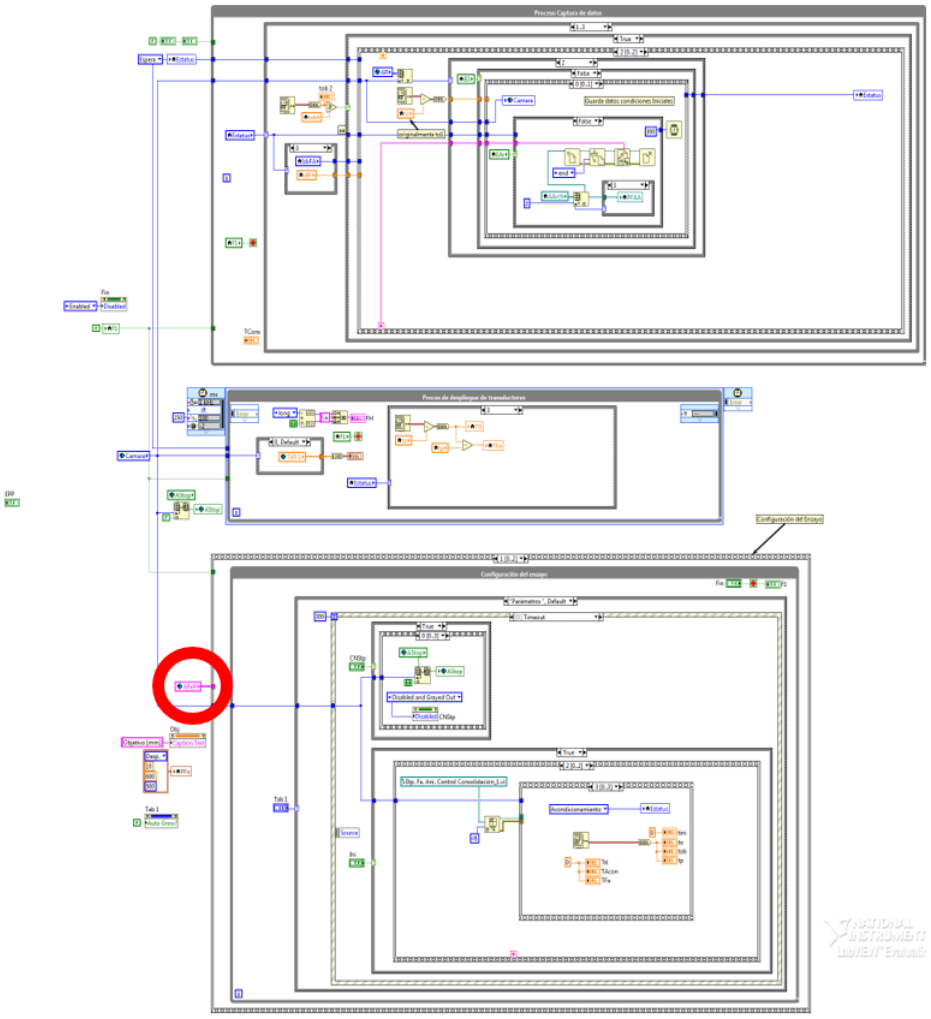

Figura 5.5: Diagrama donde se usará principalmente el módulo desarrollado, indicado dentro de un círculo rojo

En la figura 5.5 se ilustra cómo fue integrado el bloque realizado para controlar la tarjeta. El software se encarga de proporcionar a las demás regiones del programa los valores que sean requeridos para las pruebas que llevará a cabo.

En la figura 5.6 se ilustra el panel capaz de interactuar con los demás elementos

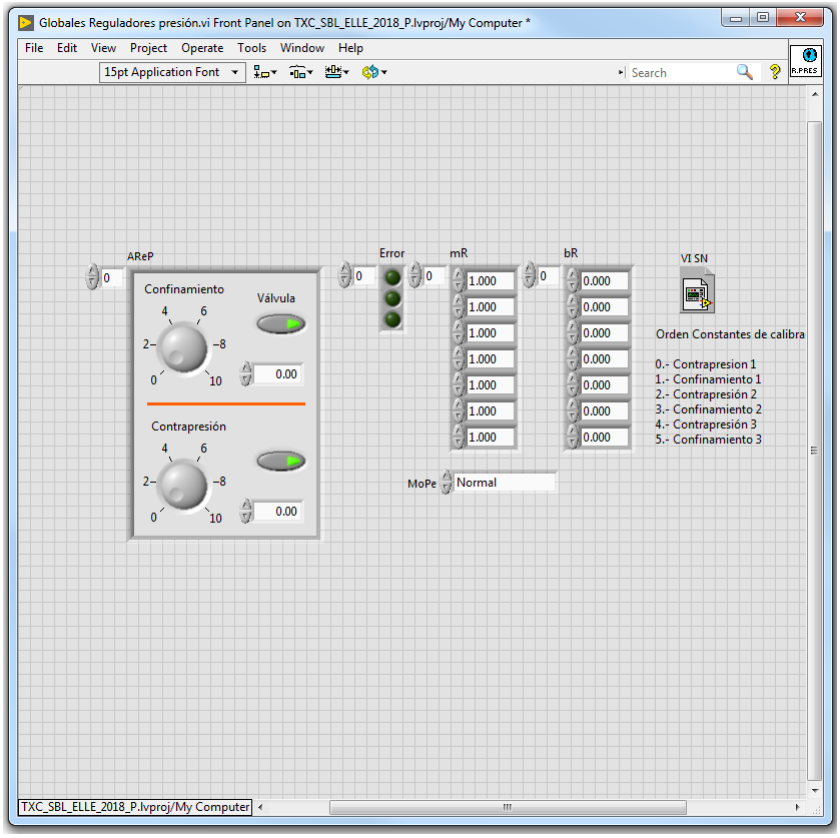

Figura 5.6: Detalle del controlador contenido en el círculo de la figura 5.5

del programa, y tal como en el capítulo anterior, esta interfaz es similar a la que interactúa con el usuario. Cabe mencionar que esta integración se llevó a cabo por el coordinador del proyecto, utilizando el código desarrollado explicado previamente.

En esta sección convergen las variables necesarias para llevar a cabo las diversas pruebas para la que se le programó, es por ello por lo que a la interfaz del capítulo anterior se le añadieron nuevos componentes para llevar a cabo este propósito.

5.4. Pruebas de comunicación

Las primeras pruebas de comunicación tras la integración únicamente consistieron en interactuar con la interfaz de usuario, se verificó que al dar clic sobre el botón de la válvula, modificara su estado y que al escribir un valor para variar la presión de

los reguladores, el valor escrito pudiera ser leído por otro módulo del software.

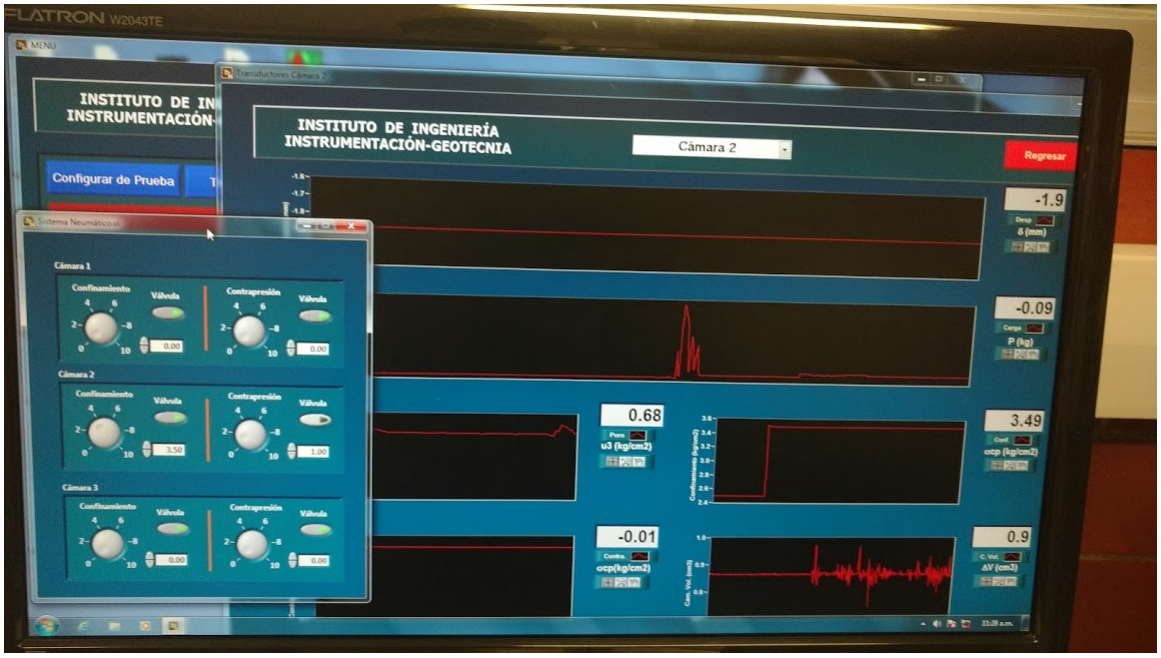

Figura 5.7: Versión inicial del software controlador

En la figura 5.7 se muestra el programa que fue utilizado para las pruebas de comunicación, dicho programa únicamente se utilizó para lo explicado previamente. Más adelante el encargado de dicho programa simplificó la interfaz y agregó la funcionalidad para llevar a cabo las pruebas correspondientes para las cuales fue diseñada la tarjeta.

5.5. Pruebas finales del sistema

Este apartado será el encargado de explicar las pruebas que fueron llevadas a cabo tras la integración y puesta en marcha del sistema, ya que a pesar de que todas las secciones realizaban las tareas para las que se programaron, se requería probar el sistema con una prueba capaz de hacer uso de la sección añadida en el presente trabajo.

A continuación, se ejemplifica la secuencia que debería llevar una prueba real, además se explicará paso a paso (gracias a la información brindada por parte del

personal del laboratorio de mecánica de suelos) las etapas en el sentido de la prueba, conociendo cómo se comporta, pero no únicamente en el modo electrónico sino en el enfoque de la mecánica de suelos.

El sistema realizado se encarga de controlar una cámara triaxial cíclica, dotándolo de la capacidad de automatizarlo prácticamente en su totalidad y se utiliza en el panel siguiente:

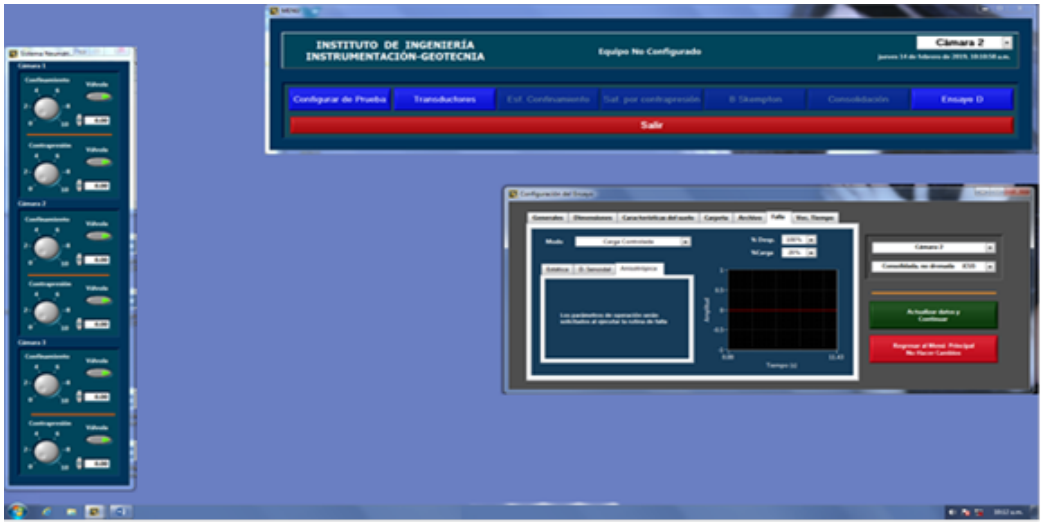

Figura 5.8: Panel principal para iniciar pruebas

En la parte izquierda de la figura 5.8, se encuentra el panel controlador de la tarjeta (reemplazo de los componentes de la figura 5.9), en la parte superior derecha se encuentra la opción para elegir el tipo de prueba y en donde se llevará a cabo la misma. Por último, en la parte media derecha, se encuentra un panel utilizado para configurar la prueba, el cual se ejecuta cuando se da clic sobre la opción de configurar prueba. Hasta este punto la única interacción que puede presentarse con el desarrollo aquí elaborado es el de poder ajustar la presión de la confinamiento o contrapresión, además de permitir la apertura y cierre de válvulas, lo cual hace que el usuario pueda configurar cualquier tipo de prueba desde la comodidad de su computadora.

Figura 5.9: Panel de control de cámaras, dos reguladores de presión y dos manómetros de carátula se reemplazaron por una interfaz gráfica en el software realizado

El usuario es quien decidirá si se utilizará o no la opción de llevar a cabo la nueva prueba, dado que la acción de apertura y cierre de las válvulas eléctricas y el control de las servoválvulas se lleva a cabo de manera manual e independiente de la sección encargada de realizar la etapa controlada de manera automática.

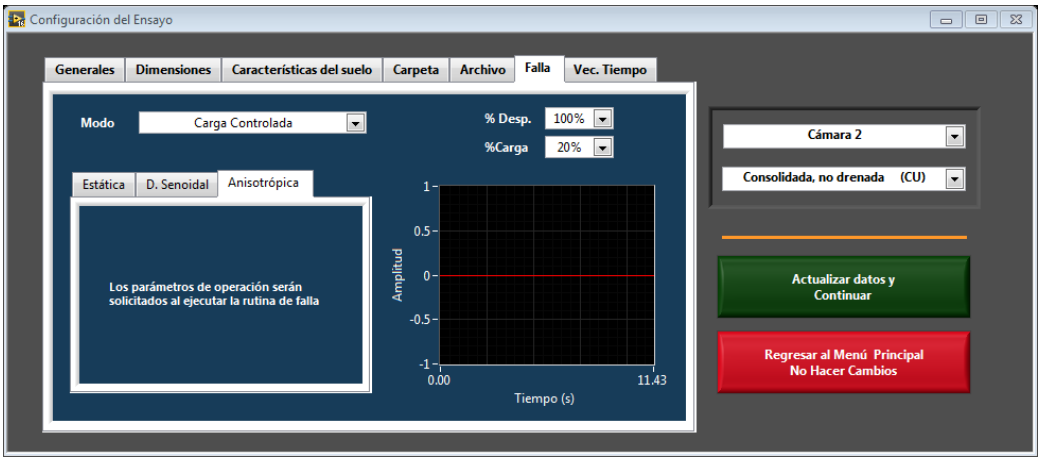

Figura 5.10: Configuración de tipo de prueba

Para poder hacer uso del sistema, se requiere especificar qué tipo de prueba se llevará a cabo (figura 5.10) y parámetros de configuración como porcentaje de desplazamiento o porcentaje de carga, ciertas medidas del equipo a actualizar, además

de seleccionar donde se guardarán los resultados obtenidos en la prueba. Este tipo de pruebas se realizar para obtener parámetros dinámicos en muestras de suelo, tales como el módulo de rigidez al corte y el amortiguamiento, donde el fallo de estas muestras se presentarán a carga controlada o desplazamiento controlado.

Figura 5.11: Sensado de los valores de la cámara ya configurada

Continuando con las pruebas, se procede a leer diversos valores en la cámara correspondiente(figura 5.11), con los cuales el programa desarrollado será capaz de interactuar y conocer el estado de la cámara. Las partes de configuración de prueba y la de transductores, únicamente se llevan a cabo en la etapa de montaje, punto en el que el espécimen de suelo se introduce en la cámara.

Dependiendo del tipo de prueba podría tener saturación por contrapresión (agua que se intenta poner a la muestra de suelo). En esta parte de la prueba se utiliza el panel para variar la presión con la finalidad de establecer un diferencial de presión, para que el agua trate de saturar los vacíos del suelo y reemplazarlos con agua, en dado caso de no saturar la muestra no se podrá seguir a la fase de consolidación. Esta prueba se apoya de un sensor volumétrico que entra al suelo y cuantifica la cantidad de agua que entra al suelo y con qué presión lo hace.

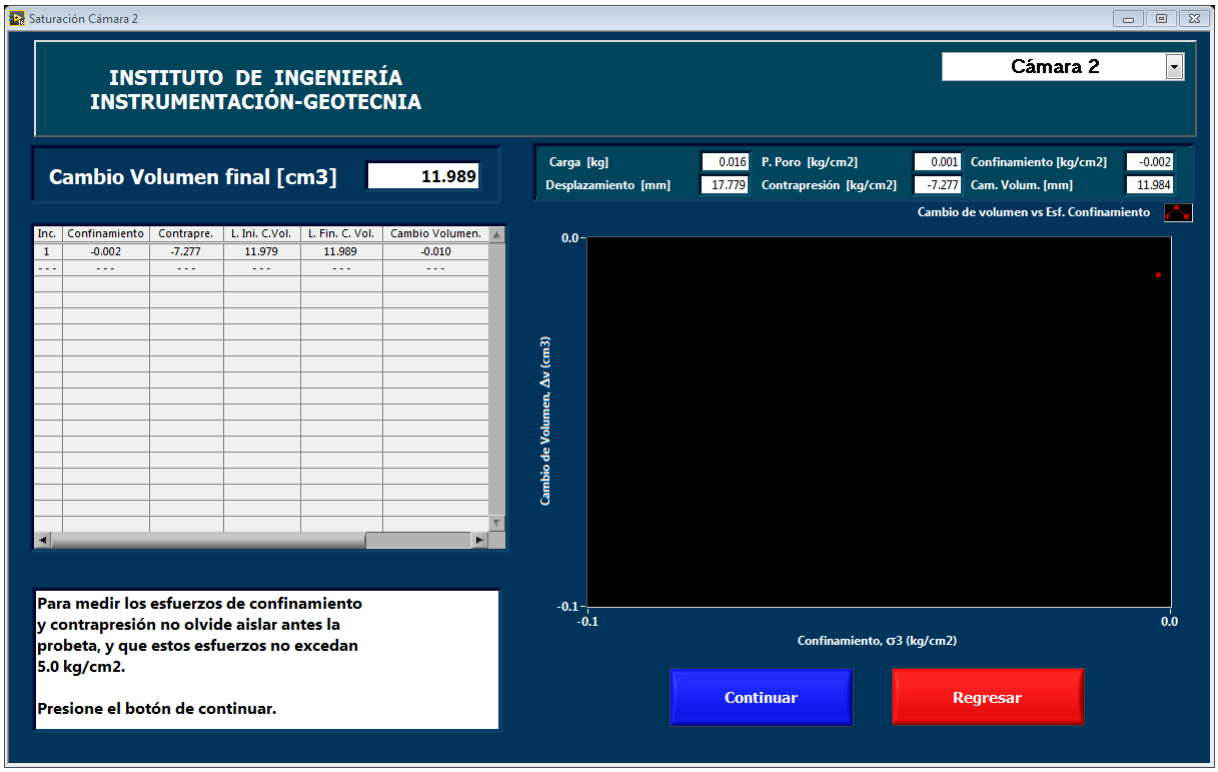

Figura 5.12: Etapa de saturación

Para llevar a la cabo la medición y saber que tan saturada se encuentra la muestra (figura 5.13), se utiliza la "B de Skempton". En esta prueba se aumenta la presión de cámara y se leen valores de los sensores, cuando se registra presión de poro se realiza un cociente de la presión de poro entre el incremento de la presión de cámara (presión confinante) si es mayor que 0.95 el suelo está saturado y listo para ser consolidado. Este proceso tiene una duración aproximada de un día, en dado caso de no saturarse se incrementa la presión y se repite la prueba. Todos los valores obtenidos, se almacenan en la ubicación previamente configurada.

Figura 5.13: Lectura y cálculo de la B de Skempton

La siguiente etapa es la de consolidación, en esta etapa se hace que el suelo expulse agua y se rigidice, al expulsarla se tiene un cambio volumétrico a razón de tiempo. Se lleva a cabo la lectura de la presión de poro para determinar cómo se va disipando conforme pasa el tiempo. Se aumenta la presión de cámara con la misma contrapresión inicial y se va drenando mediante el drene inferior. En la gráfica de la figura 5.14 se visualiza el comportamiento de esta prueba, dependiendo la forma que tenga se aplican ciertos criterios para determinar si ya se encuentra consolidada.

En la figura 5.14 también se muestra la curva de compresibilidad, la cual varía cuando se sube la presión de cámara mientas la presión de confinamiento se mantiene. Se lee el valor aplicado en la cámara, transcurridas veinticuatro horas se verifica si las curvas ya están definidas. En caso contrario se sigue aumentando la presión hasta que la prueba esta lista, en todos los casos se utiliza la misma contrapresión (esta etapa dura mínimo 5 días dependiendo del tipo de suelo analizado). Finalmente, y al tener el último esfuerzo efectivo se prosigue a la etapa de ensaye.

Figura 5.14: Etapa de consolidación

Finalizada la etapa de consolidación, se prosigue a dar inicio a la prueba de ensaye, la cual según se comentó, tiene una duración de aproximadamente un día. Dicha prueba se efectúa de la siguiente forma: se va variando el desplazamiento mientras que la presión de poro va subiendo y bajando. Cabe destacar que antes se realizaba de manera manual dicha prueba, con largas jornadas de trabajo, mediante un desplazamiento manual de la cámara (la carga se iba modificando manualmente), se tenía una lectura puntual y no continua, ya que se aumentaba y de inmediato se leían valores en los manómetros. Afortunadamente el sistema actual ahora permite tomar muchos puntos en cada lapso gracias a diversas actualizaciones que ha ido recibiendo el sistema.

Una mejora añadida que posee el equipo gracias a este desarrollo es que pueden realizar pruebas de consolidación anisótropa (figura 5.15), las cuales se efectúan al ir variando con el tiempo la presión confinante y la carga axial, provocando que ya no se mantenga constante durante la etapa de consolidación. Esto se realiza a razón de un coeficiente de la presión horizontal entre la presión vertical.

Es importante destacar que esta condición es de alta relevancia ya que es una con-

dición representativa de cómo se consolida el suelo de manera natural, dado que los centímetros del suelo se van formando de capas muy pequeñas. Esto es posible gracias a que se presenta un mayor esfuerzo vertical que horizontal. El equipo reproduce el fenómeno de consolidación en condiciones normales a las que podría estar sometido.

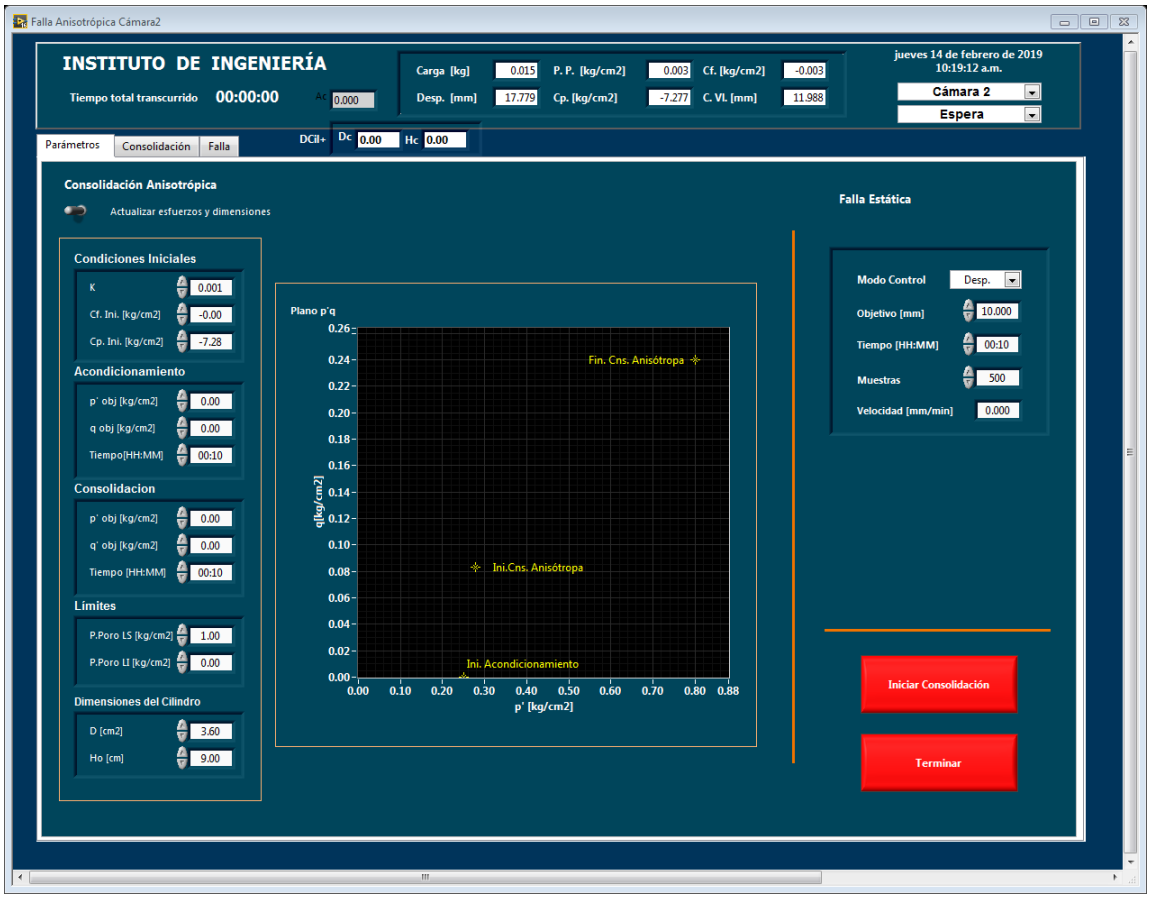

Figura 5.15: Etapa de consolidación anisótropa

Finalmente, el encargado de operar el equipo y llevar a cabo las pruebas, es capaz de interpretar los datos obtenidos en la prueba. Ya que la última prueba realizada se lleva a cabo de manera automática y solo se requiere visualizar cada lapso el proceso en el que se encuentra.

Capítulo VI

Conclusiones

Con la elaboración de este trabajo se concluye que es posible llevar a cabo la actualización de equipos utilizados para pruebas de mecánica de suelos, los cuales, sin importar su antigüedad, son capaces de poder ser controlados electrónicamente prácticamente en su totalidad, reduciendo el esfuerzo humano para su operación en las pruebas que se lleven a cabo ya que será posible realizar las pruebas que sean necesarias en el sistema actualizado desde la comodidad de su computadora.

Además, se proporciona al sistema una actualización destacable, capaz de realizar cierto tipo de pruebas, de las no se tenía forma de llevar a cabo, no debido a las limitaciones del equipo en cuestión sino por la complejidad de estas, ya que se requeriría efectuar una serie procesos manuales por parte del operador de forma iterativa y por más de veinticuatro horas continuas, sin la posibilidad de tener intervalos de descanso y con una tolerancia mínima en el momento de llevar a cabo las mediciones propias de la prueba.

Una de las ventajas de este desarrollo es que tiene la posibilidad de que se le agreguen nuevos dispositivos para poder actualizar más cámaras triaxiales que comparten ubicación con la cámara ya actualizada, todo esto gracias a que el firmware siempre será el mismo para todas las nuevas tarjetas controladoras que se añadan, necesitando como única configuración el cambio del número de dispositivo en la tarjeta. Sobre el software tampoco se requiere realizar una adaptación tan exhaustiva, dado que su interfaz ya cuenta con los controles visuales necesarios para recibir dicha actualización.

Figura 6.1: Antes de actualizar la cámara triaxial

En la figura 6.1 se presenta una gráfica que podía generarse inicialmente en el sistema de manera automática solo era posible establecer una condición isótropa en la etapa de consolidación, aplicando una presión de cámara y contra presión constante por 24 horas. Al actualizar la cámara con el desarrollo presentado en esta tesis, fue posible llevar más allá dicha prueba, dotándola de la capacidad de realizar una consolidación anisótropa, en cual se muestra en la figura 6.2.

A la izquierda se muestra el panel elaborado para llevar a cabo el control de la presión y las válvulas, que es controlado por LabVIEW. Y a la derecha las diversas gráficas por las que la prueba va tomando su forma. Por ejemplo, la presión de poro, confinamiento y finalmente los resultados de la prueba anisótropa, la cual

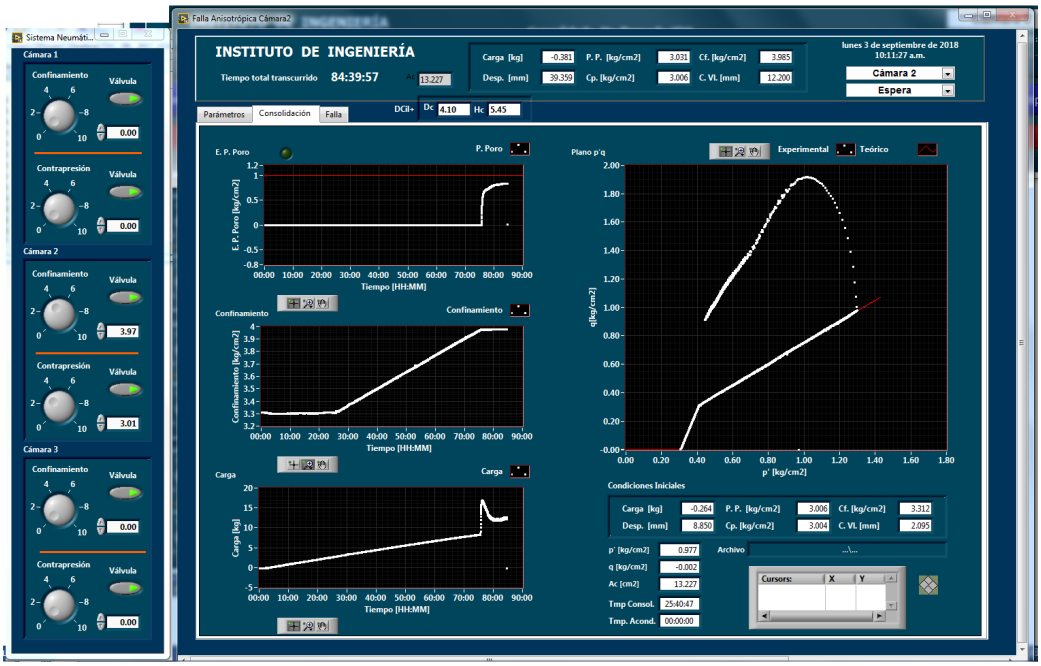

Figura 6.2: Después de actualizar la cámara triaxial

representa un hito en la mecánica de suelos del laboratorio de geotecnia, ya que tanto la complejidad, esfuerzo y tiempo que requería la prueba para llevarla a cabo impedía su desarrollo. En la imagen se muestra una duración de esta en un tiempo de 84 horas con 39 minutos, lo cual no era posible llevar a cabo de manera manual, ya que para llevar a cabo esta prueba es necesario variar con el tiempo la presión confinante, mientras que la carga axial, no se mantiene constante durante la etapa de consolidación

Finalmente, y gracias a la elaboración de este proyecto, fue posible utilizar los conocimientos obtenidos en la carrera de Ingeniería en Computación para ayudar a otra rama de la ingeniería a actualizar un sistema para tener la capacidad de llevar a cabo una serie de pruebas que anteriormente no era capaz de llevar a cabo, aun si se invirtiera un gran número de personas para realizarla.

Bibliografía

[1] *DACxx6xT Dual 16-, 14-, 12-Bit, Low-Power, Voltage-Output DACs With 2.5-V*. Texas Instruments. URL http://www.ti.com/general/docs/lit/getliterature.tsp?baseLiteratureNumber=slase61&fileType=pdf. Rev. Octubre 2015. Consultado Marzo, 2019.

[2] *LMx58-N Low-Power, Dual-Operational Amplifiers*. Texas Instruments. URL http://www.ti.com/lit/ds/symlink/lm158-n.pdf. Rev. Diciembre 2014. Consultado Marzo, 2019.

[3] *Low Power Half-Duplex RS-485 Transceivers*. Max Linear. URL https://www.maxlinear.com/ds/sp483-sp485.pdf. Rev. Junio 2011. Consultado Enero, 2019.

[4] *RealTerm: Serial/TCP Terminal*. Realterm. URL https://sourceforge.net/projects/realterm/. Actualizado 13-10-2017, consultado Febrero, 2019.

[5] *The RS-485 Design Guide*. Texas Instruments. URL http://www.ti.com/lit/an/slla272c/slla272c.pdf. Rev. Oct 2016. Consultado Enero, 2019.

[6] *Serial Instrument Control Tutorial*. National Instruments. URL http://www.ni.com/tutorial/2897/en/. Publicado: feb 27, 2019.

[7] *Ultra High Resolution electro-pneumatic closed-loop proportional pressure control valves*. Proportion Air. URL https://proportionair.com/electronic-pressure-regulators/qpv. Rev. Mayo 2016. Consultado Febrero, 2019.

[8] Eulalio Juárez Badillo y A.R. Rodríguez. *Mecánica de suelos*. Nº v. 1 en Ingeniería civil y arquitectura. Limusa, 1974. ISBN 9789681800697. URL https://books.google.com.mx/books?id=3OPOaDHQC8wC. [Online; consultado Enero, 2019].

[9] E.G. Breijo. *Compilador C CCS y Simulador Proteus para Microcontroladores PIC*. Marcombo, 2012. ISBN 9788426718648. URL https://books.google.com.mx/books?id=k8vMlKuRAyUC.

[10] Inc. Custom Computer Services. Advantages of the ccs c compiler. URL http://www.ccsinfo.com/content.php?page=compilers. [Online; consultado 19 de Febrero, 2019].

[11] IEEE. Ieee standard glossary of software engineering terminology. 1990. URL http://www.informatik.htw-dresden.de/~hauptman/SEI/IEEE_Standard_Glossary_of_Software_Engineering_Terminology%20.pdf. [Online; consultado 19 de Febrero, 2019].

[12] IINGEN. Mecánica de suelos. 2019. URL http://www.iingen.unam.mx/es-mx/Investigacion/Laboratorios/Paginas/MecanicaDeSuelos.aspx. [Online; consultado Febrero, 2019].

[13] Alejandro Jiménez Hernández. *Apuntes de compiladores*. División de Ingeniería Mécanica y Electricidad, Departamento de Ingeniería en Computación, 2010. URL http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/10230/7J%20APUNTES%20DE%20COMPILADORES_OCR.pdf?sequence=1. [Online; consultado Marzo, 2019].

[14] Melody svet hubdy. Rs232. 2019. URL https://cdn.myshoptet.com/usr/www.melodyshop.sk/user/shop/detail_alt_1/129732_fbt-usb-rs485.gif?59672319. [Online; consultado Abril 13, 2019].

[15] Memoria (proceso). Memoria (proceso) — Wikipedia, la enciclopedia libre. 2019. URL https://es.wikipedia.org/wiki/Memoria_(proceso). [Online; consultado Enero 29, 2019].

[16] Microchip. Mplab® x integrated development environment (ide). URL https://www.microchip.com/mplab/mplab-x-ide.

[17] Microchip. Pic18f2420/2520/4420/4520 data sheet. 2004. URL http://ww1.microchip.com/downloads/en/devicedoc/39631a.pdf. [Online; consultado 19 Febrero, 2019].

[18] Paul Myers. Interfacing using serial protocols using spi and i2c. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.460.2531&rep=rep1&type=pdf. [Online; consultado 19 de Febrero, 2019].

[19] F.E.V. Pérez y R.P. Areny. *Microcontroladores: fundamentos y aplicaciones con PIC*. Alfaomega. Marcombo. Marcombo, 2007. ISBN 9788426714145. URL https://books.google.com.mx/books?id=ODenKGOHMRkC. [Online; consultado Marzo, 2019].

[20] Carmelino Rivera Constantino, Rigoberto y Zea Constantino. *Notas sobre los fundamentos de la mecánica de suelos*. IINGEN, 2004. URL http://www.ingenieria.unam.mx/~especializacion/camposdisciplinarios/geotecnia/fundamentos_mecanica_suelos.pdf. [Online; consultado Marzo, 2019].

[21] Esteban Sáez. *Notas sobre los fundamentos de la mecánica de suelos*. Departamento de Ingeniería Estructural y Geotécnica, 2010. [Online; consultado Marzo, 2019].

[22] Tag-connect. Tag-connect. 2019. URL http://www.tag-connect.com/sites/default/files/images/ddblock/1tag.jpg. [Online; consultado Abril 13, 2019].

[23] Tag-Connect. Tag connect explained. 2019. URL http://www.tag-connect.com. [Online; consultado Febrero, 2019].

[24] J.R.L. Vizcaíno y J.P. Sebastiá. *Labview : entorno gráfico de programación*. Marcombo, 2011. ISBN 9788426716965. URL https://books.google.com.mx/books?id=ZFQua3-eeQEC. [Online; consultado Marzo, 2019].