



**INSTITUTO TECNOLÓGICO  
JOSÉ MARIO MOLINA  
PASQUEL Y HENRÍQUEZ  
ZAPOPAN**

**REPORTE FINAL DE RESIDENCIAS PROFESIONALES  
JULIO 2021**

Miguel Ángel Pérez Bautista

16011804

Integración de un sistema de monitoreo y control de un banco de pruebas de máquinas eléctricas

Coordinación de ingeniería en electrónica

Asesor interno: Ing. Jesús Ramón Sosa Beltrán

Asesor externo: Dr. Noé Villa Villaseñor

Camino Arenero No. 1101 Col. El Bajío 45019 Zapopan Jalisco, México.

## Agradecimientos

En primer lugar le agradezco a Dios por darme la vida, por darme el entusiasmo de aprender, por darme las ganas de conocer nuevas cosas y por llenar mi cabeza de curiosidad.

Ya que es ahí, donde nace el aprendizaje y el conocimiento, desde la curiosidad.

Le agradezco a mi familia por siempre brindar el apoyo necesario para mis estudios y por alentarme en el camino.

Le agradezco a todos los profesores que conocí a lo largo de mis estudios, por compartirme de sus conocimientos y experiencias que me fueron de utilidad para forjar mi pensamiento que hoy en día tengo, y que me ha ayudando bastante para crecer como persona.

Fueron muchas personas que estuvieron conmigo a lo largo de este camino; familia, profesores, compañeros, a todos ellos les doy las gracias por formar parte de esto.

**¡GRACIAS!**

# Resumen

# Índice

<b>1. Tabla de datos de residencia profesional</b>	<b>7</b>
<b>2. Carta de aceptación</b>	<b>8</b>
<b>3. Carta de finalización</b>	<b>9</b>
<b>4. Introducción</b>	<b>10</b>
<b>5. Justificación</b>	<b>11</b>
<b>6. Objetivos</b>	<b>12</b>
6.1. Objetivo General . . . . .	12
6.2. Objetivos específicos . . . . .	12
<b>7. Caracterización del lugar de trabajo</b>	<b>13</b>
<b>8. Alcances y limitaciones de proyecto</b>	<b>14</b>
<b>9. Fundamento teórico</b>	<b>15</b>
9.1. Sistemas embebidos . . . . .	15
9.1.1. Componentes de un sistema embebido . . . . .	15
9.1.2. Arduino . . . . .	17
9.1.3. Raspberry Pi . . . . .	18
9.1.4. ESP32 . . . . .	20
9.2. Sensores Industriales . . . . .	21
9.2.1. Encoder . . . . .	21
9.2.1.1. Funcionamiento . . . . .	21
9.2.1.2. Tipos de encoder . . . . .	22
9.2.2. Sensor de par y torsión . . . . .	23
9.2.2.1. Sensores de reacción . . . . .	24
9.2.2.2. Sensores de torque rotatorios . . . . .	24
9.2.2.3. Rotatorios de contacto . . . . .	25
9.2.2.4. Rotatorios de no contacto . . . . .	25
9.3. Protocolos de comunicación industrial . . . . .	26
9.3.1. DeviceNet . . . . .	27
9.3.2. Modbus TCP/IP . . . . .	27
9.3.3. PROFINET . . . . .	28
9.3.4. PROFIBUS . . . . .	29
9.3.5. EtherCAT . . . . .	30
9.4. Interfaz gráfica de usuario . . . . .	31
9.4.1. HMI (Human Machine Interface) . . . . .	31
9.4.1.1. Usos de la HMI . . . . .	31
9.4.1.2. HMI y SCADA . . . . .	32
9.4.1.3. Ventajas de la HMI . . . . .	32
9.4.1.4. El futuro de la HMI . . . . .	33
9.5. Servomotores Industriales . . . . .	34

9.5.0.5. Usos del servomotor . . . . .	34
9.5.0.6. Funcionamiento . . . . .	34
9.5.0.7. Componentes de un servomotor . . . . .	35
9.5.0.8. Ventajas y desventajas de servomotores . . . . .	36
9.5.0.9. Aplicaciones . . . . .	37
<b>10. Desarrollo de actividades . . . . .</b>	<b>38</b>
10.1. Cronograma de Actividades . . . . .	38
10.2. Definición de proyecto . . . . .	39
10.2.1. Modo Motor . . . . .	39
10.2.2. Modo Generador . . . . .	40
10.2.3. Primeras ideas . . . . .	40
10.3. Reconocimiento de dispositivos . . . . .	42
10.3.1. Kollmorgen Servodrive y Servomotor . . . . .	42
10.3.2. Sensor de velocidad y torque Futek . . . . .	44
10.3.2.1. Dato de velocidad . . . . .	45
10.3.2.2. Dato de torque . . . . .	46
10.3.3. Freno industrial Warner Electric . . . . .	46
10.4. Propuestas de controlador . . . . .	49
10.4.1. Raspberry Pi . . . . .	49
10.4.2. Arduino . . . . .	51
10.5. Propuesta para GUI . . . . .	55
10.6. Diagrama de bloques . . . . .	57
10.7. Desarrollo de comunicación Kollmorgen Modbus . . . . .	58
10.7.1. Lectura del manual de usuario . . . . .	58
10.7.2. Teoría básica Modbus TCP/IP . . . . .	60
10.7.3. Pruebas de comunicación Modbus TCP IP . . . . .	62
10.7.4. Identificación de registros a implementar en proyecto . . . . .	67
10.7.5. Arduino Modbus TCP/IP . . . . .	69
10.8. Sensor de torque Futek TRS605 . . . . .	73
10.8.1. Dato Velocidad . . . . .	73
10.8.2. Dato torque . . . . .	74
10.9. Freno Industrial Warner Electric . . . . .	76
10.10 Desarrollo de GUI . . . . .	78
<b>11. Resultados . . . . .</b>	<b>79</b>
11.1. Inicialización de aplicación y pantalla principal . . . . .	79
11.1.1. Carpeta de recopilación de datos . . . . .	81
11.2. Modo Motor . . . . .	83
11.3. Modo Generador . . . . .	90
11.3.1. Configuración en pestaña “Service motion” . . . . .	91
11.3.2. Cargar archivos txt file . . . . .	93
11.3.3. Pestaña “Monitor Modbus” . . . . .	94

# Índice de figuras

1.	Carta de aceptación residencia profesional	8
2.	Plano del laboratorio de eficiencia energética CIATEQ Zapopan	13
3.	Familia Arduino	17
4.	Pinout GPIO Raspberry Pi 3 B+	18
5.	Raspberry Pi 3 B+	19
6.	Diagrama de bloques de funciones de ESP32	20
7.	ESP-32	20
8.	Partes de un encoder	22
9.	Sensor de torque Futek	23
10.	Sensor de reacción	24
11.	Sensor de torque rotatorio	24
12.	Sensor del tipo rotatorio de contacto	25
13.	Sensor del tipo rotatorio de no contacto	25
14.	Logo DeviceNet	27
15.	Red Modbus TCP/IP	27
16.	Logo de PROFINET	28
17.	Logo de Profibus	29
18.	Logo de EtherCAT	30
19.	Ilustración de una HMI	33
20.	Control de lazo cerrado de un servomotor	35
21.	Partes externas del servomotor	36
22.	Servomotores industriales	37
23.	Cronograma general del proyecto de residencia	38
24.	Diagrama a bloques modo motor	39
25.	Diagrama a bloques modo generador	40
26.	Dispositivos Kollmorgen en banco de pruebas	42
27.	Software Kollmorgen WorkBench	43
28.	Sensor FUTEK TRS605	44
29.	Terminales involucradas para dato velocidad	45
30.	Señales cuadráticas de encoder	45
31.	Terminales involucradas para dato torque	46
32.	Controlador y freno industrial Warner Electric	47
33.	Tabla lógica de control Freno Warner Electric	48
34.	Red de comunicación Raspberry Pi	50
35.	Arduino Uno con Arduino Ethernet Shield	52
36.	Comparativa de capacidad de memoria arduino	53
37.	Arduino Mega junto con Ethernet shield for arduino	53
38.	Señal diferencial y señal single ended	54
39.	ADC ADS1115	54
40.	Aplicación bajo entorno MyOpenLab	55
41.	Logo Csharp y Microsoft NET	56
42.	Diagrama de bloques del proyecto	57
43.	Tabla de parámetros resumida	58
44.	Descripción de parámetro detallada	59
45.	Red Modbus TCP/IP	60

46.	Trama de comunicación modbus TCP/IP . . . . .	60
47.	Códigos de funciones para Modbus TCP/IP . . . . .	61
48.	IPs establecidas para el proyecto . . . . .	62
49.	Red local para realizar prueba de comunicación . . . . .	62
50.	Respuesta al Ping por parte de servodrive . . . . .	63
51.	Configuración para leer DIN MODE . . . . .	63
52.	Respuesta al leer registro 122 DIN MODE . . . . .	64
53.	Tráfico de comunicación . . . . .	64
54.	Configuración para escritura de registro . . . . .	65
55.	Tráfico de datos en Modbus Poll . . . . .	65
56.	Tráfico de datos mediante Wireshark . . . . .	66
57.	Registros de memoria para la configuración del movimiento . . . . .	67
58.	Trama diseñada para el envío de paquete de configuración . . . . .	67
59.	Registros servodrive para monitoreo en GUI . . . . .	67
60.	Trama de datos para monitorear en GUI . . . . .	68
61.	Registros de control de movimiento y habilitación . . . . .	68
62.	Funciones Modbus que soporta librería . . . . .	69
63.	Leer registro 1055 (DRV.HWNABLE) . . . . .	69
64.	Escribir en registro 758 (SM.V1) . . . . .	69
65.	Diagrama de red implementada para el proyecto . . . . .	70
66.	Parte del código de control para lectura de registros Servodrive . . . . .	70
67.	Parte del código de la escritura de registros servodrive . . . . .	71
68.	Bloque de código optimizado para lectura de registro modbus . . . . .	72
69.	Parte del código del encoder Futek para obtener dato de velocidad . . . . .	73
70.	Registros del microcontrolador AtMega2560 para interrupciones . . . . .	74
71.	Conexionado del módulo ADS115 . . . . .	74
72.	Configuración e inicialización del módulo ADS1115 . . . . .	75
73.	Obtener lectura de la señal analógica . . . . .	75
74.	Filtro mediante software para estabilizar lecturas de señal . . . . .	75
75.	Tabla lógica de control Freno Warner Electric . . . . .	76
76.	Circuito de conversión nivel lógico 5V - 12V . . . . .	76
77.	Definición de salidas para Freno Warner Electric . . . . .	77
78.	Código implementado para el Freno Warner Electric . . . . .	77
79.	Primer GUI beta del proyecto de residencia . . . . .	78
80.	Icono de la aplicación . . . . .	79
81.	Pantalla principal de la aplicación . . . . .	79
82.	GUI se conectó satisfactoriamente con sistema de control . . . . .	80
83.	GUI no logró conectarse con el sistema de control . . . . .	80
84.	Habilitación de modos de operación banco de pruebas . . . . .	80
85.	Selección de modo de prueba . . . . .	81
86.	Carpeta general “Datos pruebas” en disco local C . . . . .	81
87.	Carpeta de sesión . . . . .	82
88.	Carpeta de sesión . . . . .	82
89.	Ventana del modo motor . . . . .	83
90.	Cambiar de modo el Freno Industrial Warner Electric . . . . .	83
91.	Inicio de prueba modo motor . . . . .	84
92.	Graficando valores de sensor Futek TRS605 . . . . .	85

93. Aviso de exportación de gráficas . . . . .	86
94. Ventana de registro de datos . . . . .	86
95. Exportación de datos . . . . .	87
96. Datos de prueba almacenados en carpeta motor . . . . .	87
97. Salida no permitida cuando la prueba esta activa . . . . .	88
98. Prueba terminada . . . . .	89
99. Ventana principal modo generador . . . . .	90
100. Bloquear acceso a Monitor Modbus . . . . .	90
101. Parámetros guardados en text file . . . . .	92
102. Ventana para buscar el archivo txt file y cargar ajustes . . . . .	93
103. Ventana principal pestaña “Monitor Modbus” . . . . .	94

# Índice de tablas

1. Datos generales de residencia profesional . . . . .	7
--	---

## 1. Tabla de datos de residencia profesional

Campus:	Instituto Tecnológico José Mario Molina Pasquel y Henríquez campus Zapopan.
Departamento/Carrera:	Ingeniería en electrónica
Título de proyecto:	Integración de un sistema de monitoreo y control de un banco de pruebas de máquinas eléctricas
Nombre alumno:	Miguel Ángel Pérez Bautista 16011804 Ingeniería en electrónica Tec MM campus Zapopan
Asesor interno:	Ing. Jesús Ramón Sosa Beltrán
Asesor externo:	Dr. Noé Villa Villaseñor
Periodo de realización:	Febrero - Julio 2021
Área:	Laboratorio de eficiencia energética en CIATEQ Jalisco
Asignatura:	Residencia profesional
Situación del proyecto:	Proyecto Nuevo

Tabla 1: Datos generales de residencia profesional

## 2. Carta de aceptación

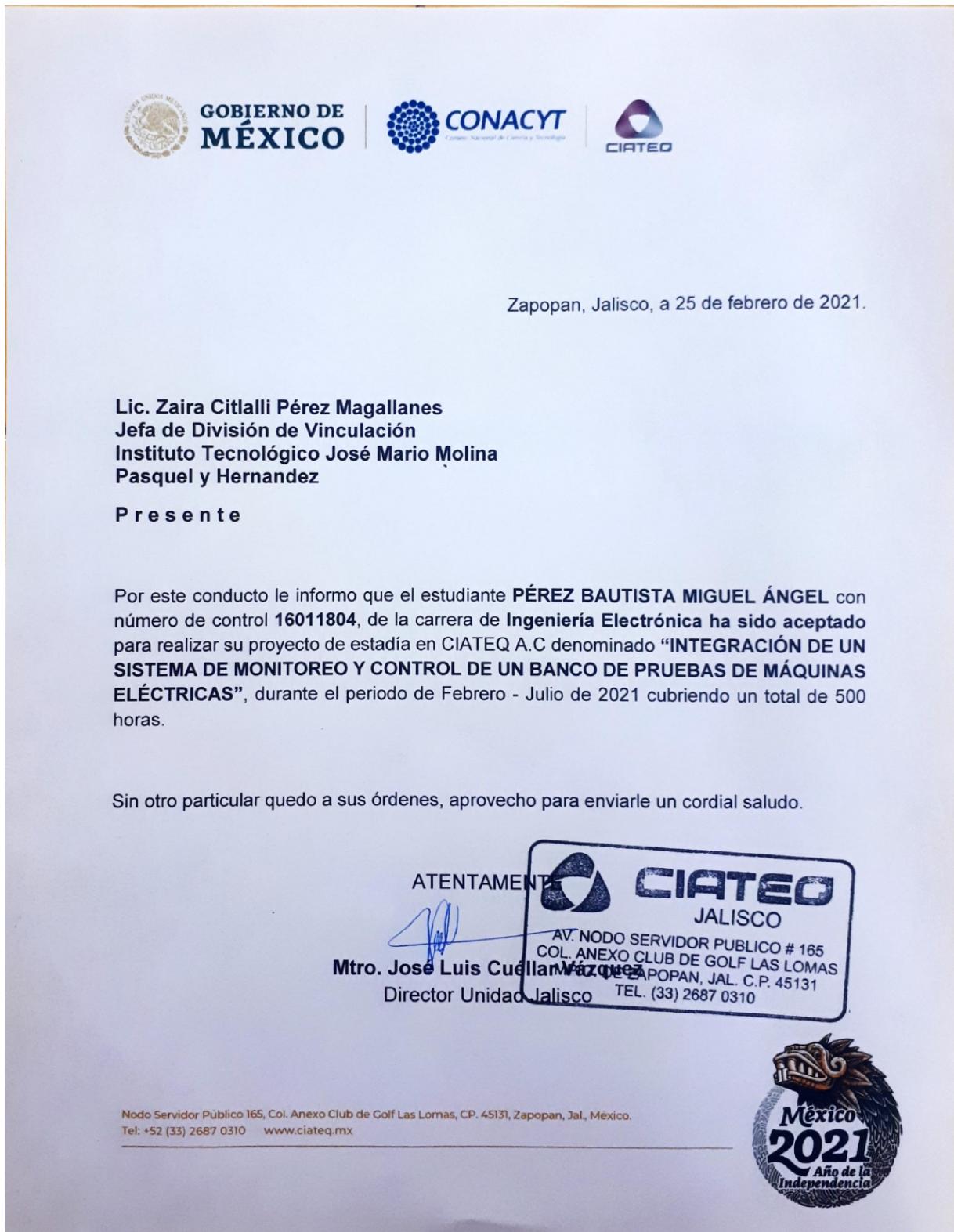


Figura 1: Carta de aceptación residencia profesional

### 3. Carta de finalización

## 4. Introducción

Una máquina eléctrica es un dispositivo capaz de transformar cualquier forma de energía en energía eléctrica o viceversa, y también se incluyen en esta definición las máquinas que transforman la energía eléctrica en la misma forma de energía pero de una forma diferente que resulta más cómoda de transportar o utilizar.

Por lo tanto, la utilidad de las máquinas eléctricas es indiscutible en la sociedad moderna desde dos vertientes; por un lado, los motores eléctricos constituyen el músculo de un sinmúmero de actividades industriales y domésticas. Por el otro, los generadores eléctricos aparecen casi en todas las formas de generación de energía eléctrica.

Los generadores transforman la energía mecánica en energía eléctrica, mientras que los motores transforman la energía eléctrica en energía mecánica girando en un eje. El motor se puede clasificar de corriente continua o motor de corriente alterna. Los transformadores y convertidores conservan la forma de la energía pero transforman sus características.

Con este avance industrial, las máquinas eléctricas comenzaron a tomar un papel importante a lo largo del tiempo en el campo industrial, se comenzó a generar electricidad a gran escala usando generadores y se dio inicio a la distribución de la electricidad desde el pueblo más chico hasta las grandes ciudades. Se comenzó a dar un nuevo giro a la forma de vivir de cada habitante sobre la tierra conforme la electricidad llegaba a sus viviendas y lugares de trabajo, puesto que ahora con electricidad se daba el inicio del desarrollo de dispositivos que funcionaran con electricidad, tal es el caso del bombillo, que una vez que su desarrollo fue completo, la vida de velas y antorchas quedó atrás en la historia de la humanidad.

Tal como pasó con los generadores de electricidad, también sucedió lo mismo para el motor eléctrico, dando paso al avance industrial para generar mayor trabajo mecánico usando una conversión de energía eléctrica a mecánica. El campo industrial comenzó a crecer exponencialmente, ya que tareas que para un humano eran difíciles de lograr, ahora para una máquina eléctrica a través de ejes de distribución de energía, engranes, poleas y cadenas, se lograba generar un trabajo mecánico eficiente y de mayor potencia, para lograr trabajos nunca antes vistos en aquella época.

Es por ello que desde aquel punto en la historia, hasta la fecha, hablar de máquinas eléctricas es un tema importante en el área de ingeniería y desarrollo de tecnología, ya que hoy en día se sigue investigando para lograr que las máquinas eléctricas sean mejores, tanto en un mejor rendimiento de potencia y eficiencia, como también usando la menor energía posible que a su vez contribuya al cuidado del medio ambiente. A nivel global se están llevando a cabo estudios para la mejora de estos dispositivos.

## 5. Justificación

Actualmente, investigadores del CIATEQ Jalisco se encuentran trabajando en el estudio e investigación de las máquinas eléctricas, por lo cual surge la necesidad de desarrollar un banco de pruebas para motores y generadores eléctricos, donde se permita al investigador de CIATEQ recolectar datos de respuesta y comportamiento.

De esta manera, los investigadores del CIATEQ Jalisco tendrán al alcancé una mesa de pruebas, donde se podrá obtener la información de respuesta que necesiten, documentarla y así poder llegar a nuevos desarrollos innovadores para máquinas eléctricas en el futuro.

## 6. Objetivos

A continuación se define el objetivo general y objetivos específicos del proyecto de residencia profesional.

### 6.1. Objetivo General

El objetivo general del proyecto es el desarrollar e integrar un sistema de monitoreo y control que permita a los investigadores de CIATEQ Zapopan ejecutar pruebas a máquinas eléctricas dando uso a los actuadores de control y sensores de una manera dinámica e interactiva a través de una aplicación de escritorio, logrando así un banco de pruebas a máquinas eléctricas automatizado

### 6.2. Objetivos específicos

Los objetivos específicos son los siguientes:

- Identificar y conocer dispositivos del sistema (banco de pruebas).
- Crear diagrama de bloques del sistema.
- Identificar comunicaciones a usar.
- Tomar lectura del manual servomotor Kollmorgen.
- Tomar lectura de comunicación Modbus.
- Identificar comandos modbus del servomotor Kollmorgen.
- hacer pruebas de comunicación Servomotor - PC.
- Analizar que plataforma de Hardware será usada como controlador (Arduino Mega, Rpi).
- Analizar que plataforma de desarrollo Software será usada para HMI/Aplicación.
- Realizar pruebas de control de dispositivos por medio del controlador.
- Comenzar con el desarrollo del programa de control del banco de pruebas.
- Comenzar con el desarrollo del programa de software (HMI/Aplicación).
- Versionado de desarrollo tanto del control como software.
- Desarrollo, pruebas y depuración entre programa de control y programa de software.
- Hacer integración completa entre control y software.

## 7. Caracterización del lugar de trabajo

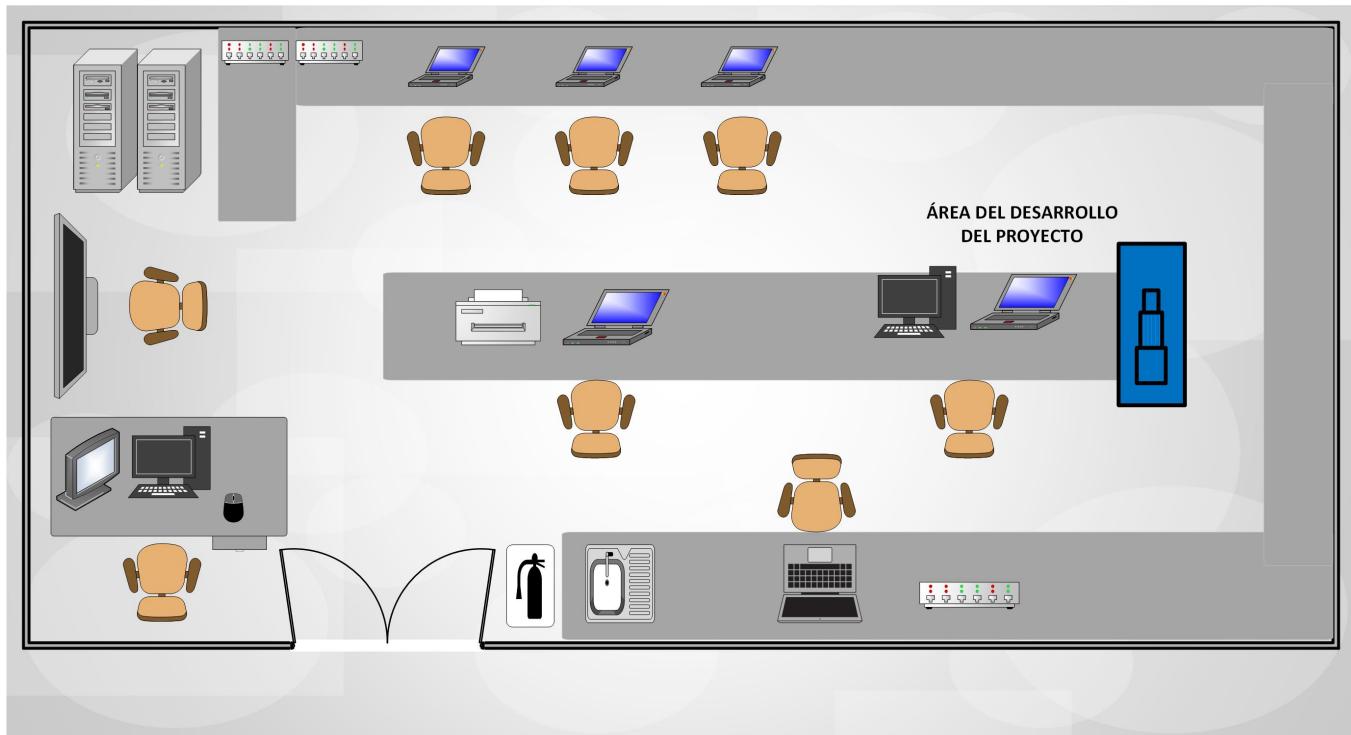


Figura 2: Plano del laboratorio de eficiencia energética CIATEQ Zapopan

## 8. Alcances y limitaciones de proyecto

El alcance del proyecto debe de permitir un buen desarrollo de programa funcional, tanto del controlador como el programa del software (interfaz de usuario), se pretende llegar a ese punto donde se pueda ingresar a los dos modos de operación del sistema y poder realizar movimientos del servomotor (en el caso del modo generador) y obtener información de sensores desplegada en el interfaz de usuario.

Por el lado del modo motor, se pretende el tomar datos de sensores y que sean visualizados de igual manera en la interfaz de usuario.

Las limitación del proyecto es el tiempo que se ha dado para desarrollar la residencia profesional, ya que se comienza de cero este proyecto y se debe de dedicar tiempo en lectura, pruebas y desarrollo, así como también tiempo a dedicar para la depuración de errores que surjan en los programas.

Por esta limitación se pretende llegar por lo menos a tener un programa de control y de software estructurados, donde el modo generador este funcionando, que se ejecute algún movimiento de servomotor y se obtengan los respectivos datos de interés.

Si existe la posibilidad de alargar fechas de entrega en residencia profesional, será posible tener también el modo motor en el sistema y completar la integración control - software.

## 9. Fundamento teórico

En esta sección se desglosan los temas teóricos para llevar a cabo el proyecto de residencia.

### 9.1. Sistemas embebidos

Un sistema embebido es un sistema de computación diseñado para realizar algunas funciones dedicadas, frecuentemente en un sistema de computación en tiempo real. Al contrario de lo que ocurre con los ordenadores de propósito general (como por ejemplo una computadora personal o PC) que están diseñados para cubrir una amplia gama de necesidades, los sistemas embebidos se diseñan para cubrir necesidades específicas.

En un sistema embebido la mayoría de los componentes se encuentran incluidos en una placa base (tarjeta de vídeo, audio, módem, etc) y muchas veces los dispositivos resultantes no tiene el aspecto de lo que se suele asociar a una computadora.

Algunos ejemplos de sistemas embebidos podrían ser dispositivos como un taxímetro, un sistema de control de acceso, la electrónica que controla una máquina expendedora o el sistema de control de una fotocopiadora entre otras múltiples aplicaciones.

Por lo general los sistemas embebidos se pueden programar directamente en el lenguaje ensamblador del microcontrolador o microprocesador incorporado sobre el mismo, o también, utilizando los compiladores específicos, pueden utilizarse lenguajes como C o C++, en algunos casos, cuando el tiempo de respuesta de una aplicación no es factor crítico, también pueden usarse lenguajes Orientados a Objetos como JAVA.

Los programas de sistemas embebidos se enfrentan normalmente a tareas de procesamiento en tiempo real.

Existen también plataformas desarrolladas por distintos fabricantes que proporcionan herramientas para el desarrollo y diseño de aplicaciones y prototipos con sistemas embebidos desde ambientes gráficos, algunos ejemplos de estas son: Arduino, mbed, Raspberry Pi, BeagleBone, Esp32, entre otros.

#### 9.1.1. Componentes de un sistema embebido

En la parte central se encuentra el microprocesador, microcontrolador, DSP, etc. Es decir, la CPU o unidad que aporta capacidad de cómputo al sistema, pudiendo incluir memoria interna o externa, un micro con arquitectura específica según los requisitos.

La comunicación adquiere gran importancia en los sistemas embebidos. Lo normal es que el sistema pueda comunicarse mediante interfaces estándar de cable o inalámbricas. Así un sistema embebido incorporará puertos de comunicaciones del tipo RS-232, RS-485, SPI, I2C, CAN, USB, WiFi, GSM, etc.

El subsistema de presentación o representación gráfica suele ser una pantalla HMI, tactil, LCD, de segmentos, etc.

Se denominan actuadores a los posibles elementos electrónicos que el sistema se encarga de controlar. Puede ser un motor eléctrico, un conmutador tipo relé, etc. El mas habitual puede ser una salida de señal PWM para control de la velocidad de motores de corriente continua.

El módulo de E/S analógicas y digitales suele emplearse para digitalizar señales analógicas procedentes de sensores, activar diodos led, reconocer el estado abierto cerrado de un conmutador o un pulsador, etc.

El módulo de reloj es el encargado de generar las diferentes señales de reloj a partir de un único oscilador principal. El tipo de oscilador es importante por varios aspectos: por la frecuencia necesaria, por la estabilidad necesaria y por el consumo de corriente requerido. El oscilador con mejores características en cuanto a estabilidad y coste son los basados en resonador de cristal de cuarzo, mientras que los que requieren menor consumo son los RC. Mediante sistemas PLL se obtienen otras frecuencias con la misma estabilidad que el oscilador patrón.

El módulo de energía, se encarga de generar diferentes tensiones y corrientes necesarias para alimentar los diferentes circuitos del sistema. Usualmente se trabaja con un rango de posibles tensiones de entrada que mediante conversores ac/dc o dc/dc se obtienen las diferentes tensiones necesarias para alimentar los diversos componentes activos del circuito.

Además de los conversores AC/DC, DC/DC, otros módulos típicos son; filtros, circuitos integrados, supervisores de alimentación, etc.

el consumo de energía puede ser determinante en el desarrollo de algunos sistemas embebidos que necesariamente se alimentan con baterías, con lo que el tiempo de uso del sistema suele ser la duración de la carga de las baterías.

Como se puede observar con la información antes presentada, un sistema embebido es un circuito diseñado para un propósito específico, quiere decir que este circuito se le puede encomendar tareas específicas para que las realice. Regularmente cuando se habla de sistemas embebidos se habla de microprocesadores y microcontroladores, que son pieza fundamental en cualquier sistema embebido en el que se planea desarrollar.

Se habla de sistemas embebidos en esta sección ya que es la parte inicial de cualquier sistema de control sea industrial o sea orientado a un placa de desarrollo de prototipos e investigación.

Por lo tanto, un sistema de control es un sistema embebido en el cual cuenta con lo necesario para poder programar tareas que se necesite que el sistema realice por nosotros, por ejemplo con un sistema de control podríamos desarrollar un programa de control que nos ayude a monitorear la cantidad de luz que tenga un cierto cuarto y se despliegue en una pantalla LCD. Otro ejemplo común sería el monitorear la temperatura y la humedad de algún jardín. Todo esto se logra dando uso de un sistema de control, que por ende es un sistema embebido.

### 9.1.2. Arduino

Arduino es una plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un microcontrolador y una serie de pines como E/S digitales y analógicas. Estos permiten establecer conexiones entre el microcontrolador y los diferentes sensores y actuadores de una manera muy sencilla (principalmente con cables dupont)

Una placa electrónica es una PCB (Printed Circuit Board, placa de circuito impreso). Las PCB son superficies planas fabricadas en un material no conductor, la cual consta de distintas capas de material conductor. Una PCB es la forma más compacta y estable para construir un circuito electrónico. Por lo tanto, la placa Arduino no es más que una PCB que implementa un determinado diseño de circuitería interna. De esta forma el usuario final no se debe de preocupar por las conexiones eléctricas que necesita el microcontrolador para funcionar, y puede empezar directamente a desarrollar las diferentes aplicaciones electrónicas que necesite.

Cuando hablamos de arduino, deberíamos de especificar el modelo concreto. Se han fabricado diferentes modelos de placas Arduino oficiales, cada una pensada como un propósito diferente y características variadas (como el tamaño físico, número de pines E/S, modelo del microcontrolador, etc). A pesar de las varias placas que existen todas pertenecen a la misma familia (microcontroladores AVR marca Atmel). Esto significa que comparten la mayoría de sus características de software, como arquitectura, librerías y documentación.

El microcontrolador que lleva la placa de Arduino Uno es el modelo Atmega328P. La "P" del final significa que este chip incorpora la tecnología "Picopower" (propiedad de Atmel) y permite un consumo eléctrico ligeramente menor comparándolo con el modelo equivalente sin "Picopower", Atmega328 (sin la P). Aunque el Atmega328P puede trabajar a un voltaje menor y consumir menos corriente que el Atmega328, ambos modelos son funcionalmente idénticos, es decir, pueden ser remplazados el uno por el otro.

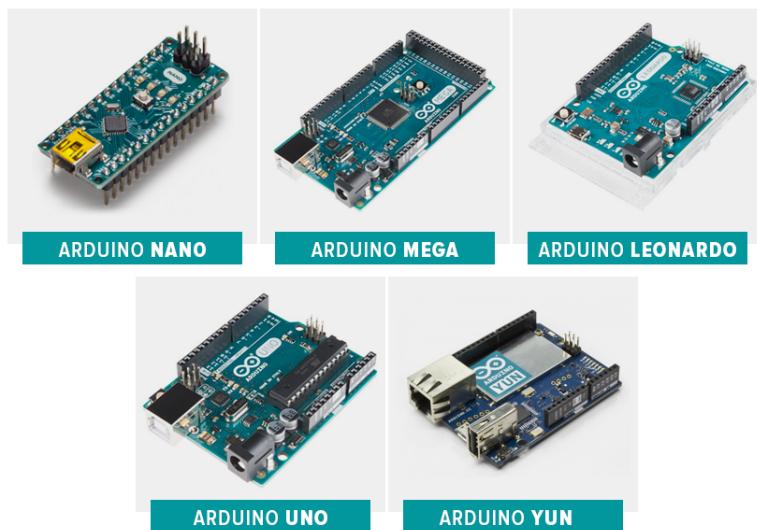


Figura 3: Familia Arduino

### 9.1.3. Raspberry Pi

Las Raspberry PI es una computadora de bajo costo y con un tamaño compacto, del tamaño de una tarjeta de crédito, puede ser conectada a un monitor o a un TV y usarse con un mouse y teclado estándar. Es un pequeño computador que corre un sistema operativo linux capaz de permitirle a las personas de todas las edades explorar la computación y aprender a programar lenguajes como scratch y Python.

Es capaz de hacer la mayoría de las tareas típicas de un computador de escritorio, desde navegar en internet, reproducir videos, manipular documentos y hasta reproducir juegos.

Además la Raspberry Pi, tiene la habilidad de interactuar con el mundo exterior, puede ser usada en una amplia cantidad de proyectos digitales, desde reproductores de música y video, detector de rostros, estaciones meteorológicas hasta cajas de aves con cámaras infrarrojas.

Raspberry es muy usada por gente dedicada al desarrollo de software, desarrollo de aplicaciones, desarrollo de proyectos y prototipos. Además de que también entra en el área de la investigación e ingeniería donde con ayuda de plataformas y hardware externo se pueden realizar proyectos muy interesantes.

Las Raspberry también es un sistema embebido, al igual que arduino, con la única diferencia de que el arduino es un microcontrolador que está listo para recibir órdenes, y una Raspberry cuenta con un microprocesador el cual funciona más como una computadora, ya que a un arduino no es posible cargarle un sistema operativo y una Raspberry PI si.

Lo interesante de la Raspberry es que tiene un sin fin de aplicaciones mayores a las de un arduino, por que se puede dar uso de Python, OpenCV, bases de datos MySQL además se tiene dentro del mismo hardware módulo bluetooth y Ethernet, entre otros.

Recursos que arduino no tiene, por lo tanto Raspberry es una buena opción si tu proyecto requiere dar uso a los recursos de hardware que Rpi tiene embebidos en su placa.

Algunas de sus especificaciones técnicas que tiene Raspberry PI 3 B+ se tienen en la siguiente imagen:

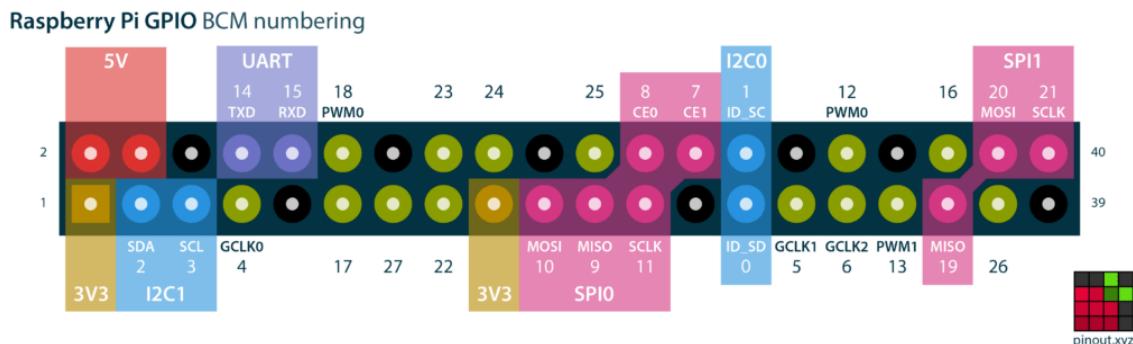


Figura 4: Pinout GPIO Raspberry Pi 3 B+

La Raspberry Pi 3 B+ cuenta con un GPIO de 40 pines, el cual permite el contacto con el mundo exterior, tanto por sensores como actuadores, en este punto es importante conocer que el GPIO de Raspberry trabaja con un nivel de 3.3 V, así que si quieres conectar sensores que operan a 5V necesitarás un conversor de niveles lógicos, el cual se recomienda el MCI00582 que es comercializado por MCI Electronics.

Debido que el procesador de la Raspberry Pi no tiene un conversor analógico a digital embedido, y si se requiere leer sensores analógicos se debe de dar uso a un conversor ADC externo, el cual el que recomienda el fabricante es el MCI01856. Rpi además cuenta con puertos de comunicación I2C, SPI y UART.

La Raspberry Pi 3B+ cuenta con conexiones tradicionales como son puertos USB, conector de red ethernet, jack de 3.5 mm , puerto HDMI, puerto para memoria microSD y un conector micro-usb para alimentación. También se destacan los puertos especiales para la cámara y la pantalla.



Figura 5: Raspberry Pi 3 B+

#### 9.1.4. ESP32

ESP32 es una serie de SoC (System on Chip) y módulos de bajo costo y bajo consumo de energía creado por Espressif Systems.

Esta nueva familia es la sucesora del conocido ESP8266 y su características más notable es que, además de WiFi, también soporta Bluetooth. ESP32 se basan en un microprocesador Tensilica Xtensa LX6 (De uno o dos nucleos) con una frecuencia de operación de hasta 240 Mhz.

Los ESP32 poseen un alto nivel de integración, en su pequeño encapsulado incluyen:

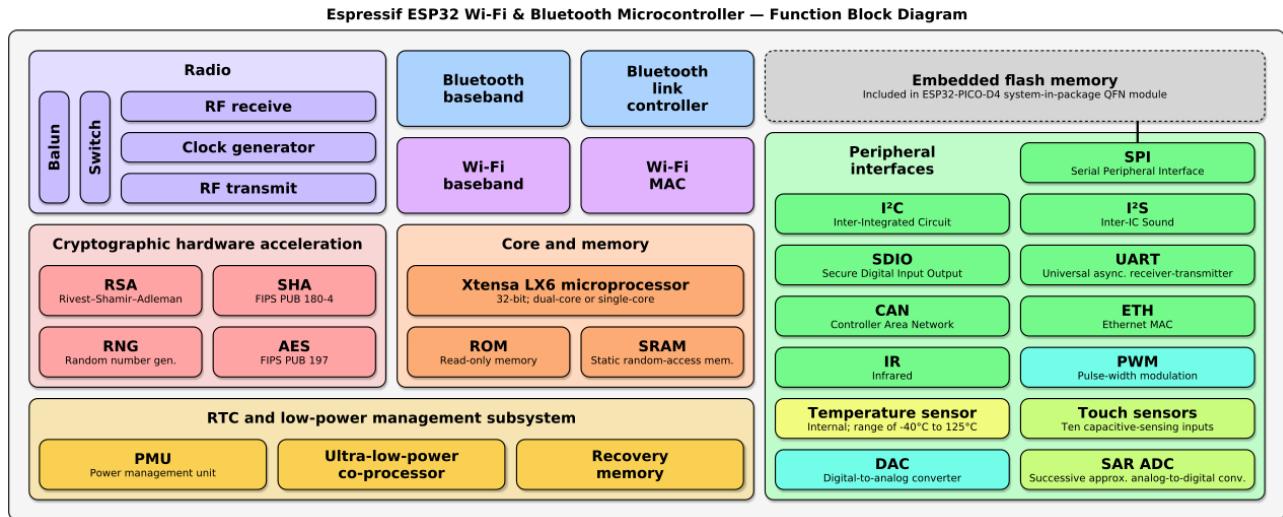


Figura 6: Diagrama de bloques de funciones de ESP32

Además logra un consumo de energía muy bajo a través de funciones de ahorro de energía que incluyen sincronización de reloj y múltiples modos de operación. Todo esto lo convierte en la herramienta ideal para tus proyectos energizados con baterías o aplicaciones IoT.

Este sistema embebido tiene gran robustez para el desarrollo de aplicaciones de media y gran escala, cuenta con el hardware y potencia necesaria para llevar a cabo tus proyectos a otro nivel.

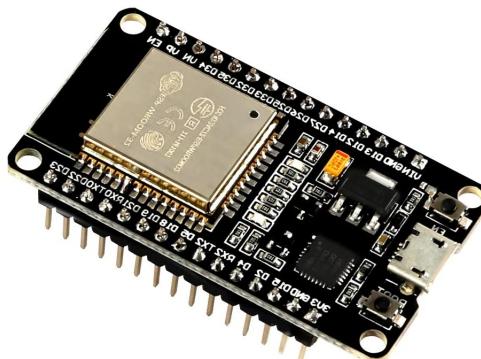


Figura 7: ESP-32

## 9.2. Sensores Industriales

Para poder controlar automáticamente un proceso de fabricación es necesario disponer de información sobre el estado del proceso. Esto se puede hacer midiendo diferentes magnitudes físicas que intervienen en el mismo. Pese a poder recibir diferentes nombres (detector, transductor, sonda), el sensor es el nombre más utilizado en control de autómatas para referirse al dispositivo que mide una magnitud física.

En general estas magnitudes físicas no tienen por que ser eléctricas, por lo que se utilizan transductores para convertir a señales eléctricas.

Un transductor convierte una señal no eléctrica en eléctrica, en la cual alguno de sus parámetros (nivel de tensión, corriente, frecuencia) contiene información sobre la magnitud de medida. Puesto a que es necesario acoplar la salida de este dispositivo transductor al sistema de control, puede ser necesario efectuar filtrado y amplificación de la señal eléctrica en el llamado circuito acondicionador o de acondicionamiento.

En esta sección se hablará de los sensores de encoder, de velocidad y torque.

### 9.2.1. Encoder

El encoder es un transductor rotativo, que mediante una señal eléctrica sirve para indicar la posición angular de un eje, velocidad y aceleración del rotor de un motor.

#### 9.2.1.1. Funcionamiento

Un encoder se compone básicamente de un disco conectado a un eje giratorio.

El disco está hecho de vidrio o plástico y se encuentra 'codificado' con unas partes transparentes y otras opacas que bloquean el paso de luz emitida por la fuente de luz (típicamente emisores infrarrojos). En la mayoría de los casos, estas áreas bloqueadas (codificadas) están arregladas en forma radial.

A medida que el eje rota, el emisor infrarrojo emite luz que es recibida por el sensor óptico (o foto-transistor) generando los pulsos digitales a medida que la luz cruza a través del disco o es bloqueada en diferentes secciones de este.

Esto produce una secuencia que puede ser usada para controlar el radio de giro, la dirección del movimiento e incluso la velocidad.

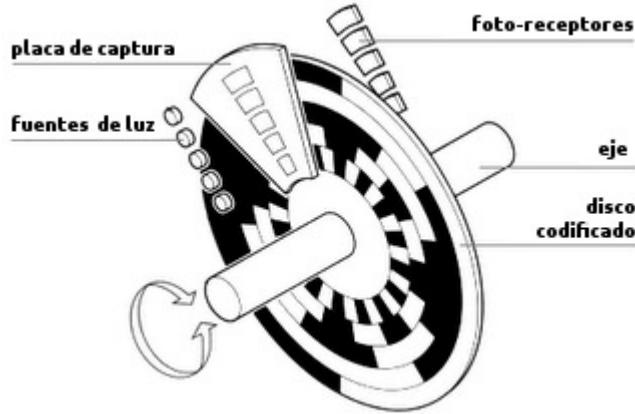


Figura 8: Partes de un encoder

### 9.2.1.2. Tipos de encoder

Una clasificación de los encoder según el tipo de información sobre la posición que genera sería:

**Incremental** Como su nombre lo indica, es un encoder que determina el ángulo de posición por medio de realizar cuentas incrementales.

Esto quiere decir que el encoder incremental provee una posición estratégica desde donde siempre comenzará la cuenta. La posición actual del encoder es incremental cuando es comparada con la última posición registrada por el sensor.

Los encoders incrementales son un tipo de encoder óptico y este en este tipo de encoder cada posición es completamente única. Un encoder incremental típico es el encoder de cuadratura.

Corresponde a un tipo de encoder incremental que utiliza dos sensores ópticos posicionados con un desplazamiento de  $\frac{1}{4}$  de ranura el uno del otro, generando dos señales de pulsos digitales desfasadas en  $90^\circ$  o en cuadratura. A estas señales de salida, se les llama comúnmente A y B. Mediante ellas es posible suministrar los datos de posición, velocidad y dirección del eje. Si se incluye la señal de referencia, se le denomina I (índice).

Usualmente, si la señal A adelanta a la señal B (la señal A toma valor lógico '1' antes que la señal B, por ejemplo), se establece el convenio de que el eje está rotando en sentido horario, mientras que si B adelanta a A, el sentido será antihorario.

**Absoluto** Se basa en la información proveída para determinar la posición absoluta en secuencia. Un encoder absoluto ofrece un código único para cada posición.

Se dividen en dos grupos: los encoders de un solo giro y los encoders absolutos de giro múltiple.

**Monovuelta** Dividen una revolución mecánica en un número determinado de pasos de medición. Tras una revolución completa, los valores de medición se repiten. El número máximo de pasos es de 8.192

**Multivuelta** No sólo registran la posición angular, sino que también cuentan las revoluciones (hasta un máximo de 4.096). La emisión de las señales se efectúa ya sea a través de una interfaz SSI o de un sistema de bus tipo CAN o Profibus.

### 9.2.2. Sensor de par y torsión

Los sensores de par y torsión miden la fuerza de torsión a la que se somete un eje durante las diferentes fases de su funcionamiento, bien sea en arranque, dinámico o parada.

Estos sensores de par y torsión se utilizan para ensayar y estudiar elementos de rotación como motores, generadores, alternadores, entre otros. También hay sensores de par estático que miden la torsión o par sin que exista rotación, por ejemplo en una llave dinamométrica.

Un transductor de par proporciona una variación mecánica en una eléctrica, en este caso una torsión se traduce en una variación de voltaje.

Los tipos de sensores de par y torsión son diferentes, pero por tecnología se dividen en estáticos o dinámicos, a su vez, estos últimos se dividen en sensores de par dinámicos sin escobillas o anillos rozantes y sensores de par con escobillas o anillos rozantes.

Los formatos son variados, aunque siempre hay un denominador común, consta de un eje instrumentado, que ha de ser intercalado entre la fuente y carga, para que el sensor de par, sea sometido a la torsión que deseamos medir.



Figura 9: Sensor de torque Futek

### 9.2.2.1. Sensores de reacción

Los sensores de reacción utilizan galgas extensiométricas en configuración de puente completo con partes inmóviles que no requieren mantenimiento. Pueden ser usados para comprobar el torque residual en sujetadores y determinar el torque para vencer los efectos de la fricción.

Su principal diferencia radica en que este tipo de sensores no gira su eje.

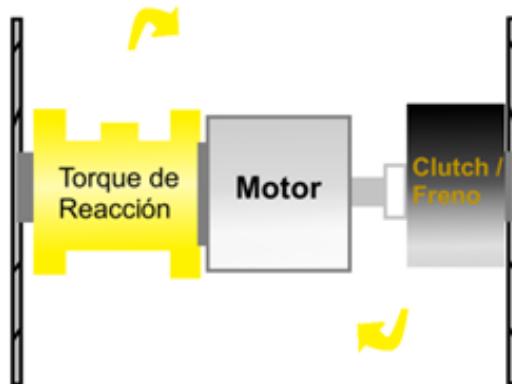


Figura 10: Sensor de reacción

### 9.2.2.2. Sensores de torque rotatorio

Los sensores de torque rotatorio ofrecen ventajas para realizar mediciones en el desempeño de herramientas y maquinaria. Se ofrecen configuraciones de montaje sin contacto y son capaces de detectar el torque dinámico en velocidades de hasta 12,000 RPM.

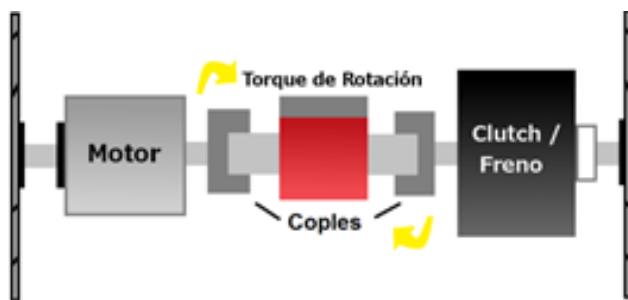


Figura 11: Sensor de torque rotatorio

### 9.2.2.3. Rotatorios de contacto

En estos sensores giratorios los elementos de medición están conectados por medio de escobillas internas, lo que no permite velocidades altas.

Sus principales características son:

- Capacidad desde 10Nm hasta 10,000 Nm.
- Velocidades hasta 3,000 RPM.

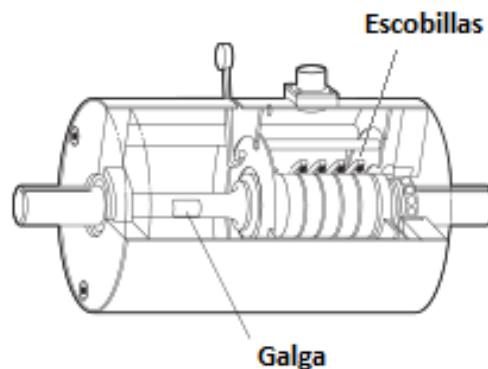


Figura 12: Sensor del tipo rotatorio de contacto

### 9.2.2.4. Rotatorios de no contacto

En estos sensores giratorios la medición de torque se realiza por medios magnéticos, por lo que, al no tener elementos de contacto, permite velocidades más grandes y no hay desgaste de las escobillas.

Sus principales características son:

- Capacidad desde 10Nm hasta 10,000 Nm.
- Velocidades hasta 12,000 RPM.

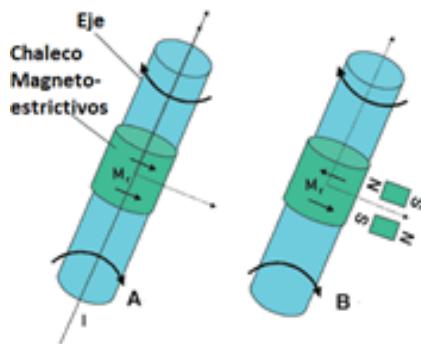


Figura 13: Sensor del tipo rotatorio de no contacto

La avanzada tecnología de los sensores basados en galgas extensiométricas es ampliamente conocida en muchas áreas de la industria por poseer:

- Alta exactitud, la cual cumple con muchos estándares de calidad en la industria automotriz y aeroespacial.
- Construcción robusta en acero inoxidable o aluminio para dar mayor resistencia.
- Garantía de fabricación para una larga duración incluso en las más rigurosas condiciones de trabajo.
- Una amplia variedad de opciones de montaje, tamaños y geometrías.
- Precio competitivo.
- Amplia variedad de capacidades.
- Alta respuesta frecuencial acoplada con diseños mecánicamente rígidos.

Estos sensores de torque son empleados en muchas aplicaciones dentro de la industria tales como médica, automotriz, aviación, aeroespacial, computación y automatización, entre otras.

### **9.3. Protocolos de comunicación industrial**

Un protocolo de comunicación industrial son un conjunto de reglas que permiten las interacciones e intercambios de datos entre varios dispositivos que forman una red.

A medida que la tecnología ha avanzado, estos van teniendo un proceso de evolución, las comunicaciones a este nivel debe de poseer unas características particulares para responder a las necesidades de intercomunicación en tiempo real. Los protocolos que se usan en la industria provienen, por un lado, de la evolución de los antiguos protocolos basados en comunicaciones serie, y, por otro lado, de la creación de nuevos estándares basados en nuevas tecnologías. Como ejemplo de evolución de los antiguos protocolos se pueden citar Modbus/TCP, DNP3, Profinet, etc.

Estos protocolos se aprovechan de las ventajas funcionales y de seguridad que ofrecen tanto Ethernet como TCP/IP para ofrecer mejores capacidades de transferencia de información en los sistemas de control. Así, la mayoría de ellos se basan en encajar la parte de datos del protocolo original en la parte de datos de una trama Ethernet.

Basa en los protocolos estándar TCP/IP, utiliza ya los bastantes conocidos hardware y software ethernet para establecer un nivel de protocolo para configurar, acceder y controlar dispositivos de automatización industrial. Ethernet/IP clasifica los nodos de acuerdo a los tipos de dispositivos preestablecidos, con sus actuaciones específicas. Ethernet/IP ofrece un sistema integrado completo, enterizo, desde la plana industrial hasta la red central de la empresa.

Uno de los objetivos principales del Ethernet/IP es que utiliza todas sus herramientas y tecnologías tradicionales, como los son los protocolos de transporte (TCP), Internet(IP) y las tecnologías de acceso y señalización de medios que se encuentran dentro de las tarjetas de interfaz de Ethernet.

### 9.3.1. DeviceNet

DeviceNet es un protocolo de comunicación usado en la industria de la automatización para interconectar dispositivos de control para el intercambio de datos. Un protocolo de comunicación es el quien permite que dispositivos individuales (arrancadores, sensores, escáner, motores, etc) comuniquen con el controlador de red. Una forma de verlo es que el término capa de aplicación implica que DeviceNet trata más con los datos de la aplicación que un nivel más bajo o un protocolo de capa que no es de aplicación.



Figura 14: Logo DeviceNet

### 9.3.2. Modbus TCP/IP

El protocolo Modbus TCP/IP es un protocolo de comunicación diseñado para permitir a que equipos industriales tales como PLCs, PCs, drivers para motores y otros tipos de dispositivos físicos de entrada/salida puedan comunicarse sobre una red Ethernet, mientras que el Modbus RTU es una representación binaria compacta de los datos adquiridos. El protocolo Modbus permite el control de una red de dispositivos, por ejemplo para que un equipo de medición de temperatura y humedad pueda comunicar los resultados a una computadora. Modbus también se usa para la conexión de un PC de supervisión con una unidad remota (RTU) en sistemas de supervisión de adquisición de datos (SCADA).

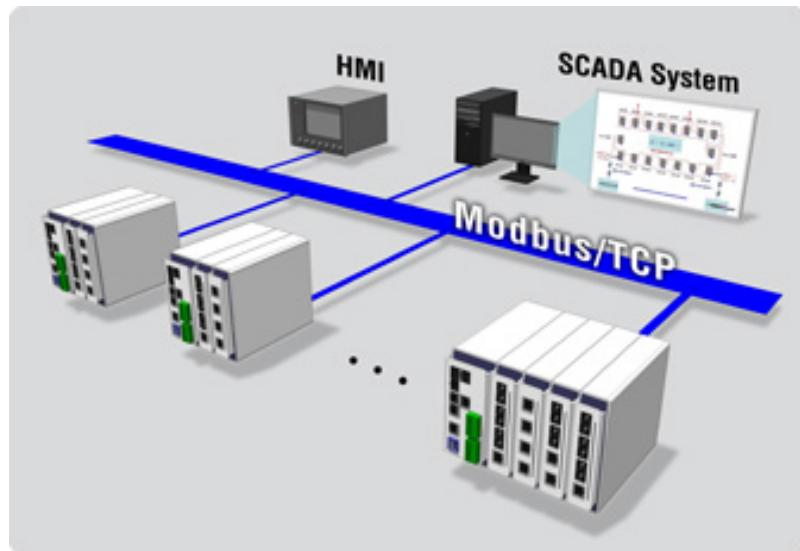


Figura 15: Red Modbus TCP/IP

### 9.3.3. PROFINET

Profinet está basado en Ethernet industrial, TCP/IP y algunos estándares de comunicación pertenecientes al mundo TI.

Entre sus características destaca que es Ethernet en tiempo real, donde los dispositivos que se comunican por el bus de campo acuerdan cooperar en el procesamiento de solicitudes que se realizan dentro del bus.

Existen varios protocolos definidos dentro del contexto PROFINET como son:

**PROFINET/CBA:** Protocolo asociado a las aplicaciones de automatización distribuida en entornos industriales.

**PROFINET/DCP:** Descubrimiento y configuración básica. Es un protocolo basado en la capa de enlace, utilizado para configurar nombres de dispositivos y direcciones IP.

**PROFINET/IO:** A veces llamado PROFINET-RT(RealTime), es utilizado para comunicaciones con periferias descentralizadas.

**PROFINET/MRP:** Protocolo utilizado para la redundancia de medios. Utiliza los principios básicos para la reestructuración de las redes en caso de sufrir un fallo cuando la red posee la topología en anillo.

**PROFINET/MRRT:** Su objetivo es dar soluciones a la redundancia de medios para PROFINET/RT.

**PROFINET/PTCP:** Protocolo de control de precisión de tiempo basado en la capa de enlace, para sincronizar señales de reloj/tiempo en varios PLC.

**PROFINET/RT:** Transferencia de datos en tiempo real.

**PROFINET/RT:** Transferencia de datos isócrono en tiempo real.



Figura 16: Logo de PROFINET

#### 9.3.4. PROFIBUS

Es un estándar de red de campo abierto e independiente de proveedores, donde la interfaz de ellos permite amplia aplicación de procesos, fabricación y automatización predial.

Profibus proporciona tres versiones diferentes de protocolos de comunicación:

**Profibus-DP:** este protocolo está optimizado para conseguir una alta velocidad de transmisión. Está especialmente diseñado para establecer la comunicación entre el controlador programable (PLC) y los dispositivos de entrada/salida a nivel campo.

**Profibus-PA:** está especialmente diseñada para conseguir una comunicación fiable a alta velocidad en ambientes expuestos a peligro de explosión.

**Profibus-FMS:** se utiliza para la comunicación a nivel célula, donde lo principal es el volumen de información y no el tiempo de respuesta.



Figura 17: Logo de Profibus

### 9.3.5. EtherCAT

Es un protocolo de código abierto para informática y tiene alto rendimiento que pretende usar protocolos Ethernet en el ambiente industrial y es uno de los protocolos más rápidos en la actualidad. Por ejemplo, con el método de transmisión de EtherCat, el paquete de Ethernet se recibe, después es copiado en el dispositivo, se interpreta y se procesa. Los medios de EtherCAT encajan bien para los ambientes industriales o de control puesto que pueden ser accionado con o sin interruptores.

EtherCAT es un estándar abierto que se ha publicado como especificación del IEC basada en la entrada del grupo de la tecnología de EtherCAT.

EtherCAT es especialmente adecuado para las aplicaciones rápidas. Entre las aplicaciones características se incluyen máquinas de embalar, máquinas de fundición y centros de mecanizado.

Las capacidades de sincronización y utilización completa del ancho de banda son muy atractivas para aplicaciones de movimiento donde se requiere la sincronización de un gran número de unidades. Ahorra gastos de instalación al eliminar tanto la topología de inicio de Ethernet como todos los commutadores, enrutadores y concentradores.

EtherCAT encaja en el espectro de capas de aplicaciones de Ethernet donde el rendimiento, la topología y el costo general de implementación son factores determinantes.



Figura 18: Logo de EtherCAT

## 9.4. Interfaz gráfica de usuario

La interfaz de gráfica de usuario o GUI (Graphic User Interface) es el entorno visual de imágenes y objetos mediante el cual una máquina y un usuario interactúan. A mediados de los años setenta, las GUI comenzaron a sustituir a las interfaces de línea de comandos, y esto permitió que la interacción con las computadoras fuera más sencilla e intuitiva.

En el campo de la ingeniería en mecatrónica las interfaces gráficas de usuario son parte fundamental de cualquier sistema desarrollado, ya que estas mismas son las que logran mantener una comunicación Humana - máquina y poder observar la adquisición de datos del sistema, como por ejemplo los valores de estado de sensores y accionamiento de ciertos actuadores, entre otros.

### 9.4.1. HMI (Human Machine Interface)

Las HMI son las siglas de Human-Machine Interface, se refieren a un panel que permite a un usuario comunicarse con una máquina, software o sistema. Técnicamente, se puede referir a cualquier pantalla que se use para interactuar con un equipo, pero se utiliza normalmente para los entornos industriales. Las HMI muestran datos en tiempo real y permiten al usuario controlar con una interfaz gráfica de usuario

En un entorno industrial, una HMI puede tener distintas formas. Puede ser una pantalla independiente, un panel acoplado a otro equipo o incluso una tablet. Da igual su aspecto; su uso principal es permitir a los usuarios visualizar los datos operativos y controlar las máquinas. Los operarios pueden usar una HMI como por ejemplo, ver qué cintas transportadoras están encendidas o ajustar la temperatura de un depósito de agua industrial.

#### 9.4.1.1. Usos de la HMI

La HMI se utiliza en una amplia gama de sectores. Es común en la fabricación de distintos productos, desde automóviles, comida y bebidas y medicina. Industrias como la de la energía, el agua, las aguas residuales, los edificios y el transporte también pueden usar HMI. Cargos como los integradores de sistemas, operarios e ingenieros y, en espacial, los ingenieros en sistemas de control de procesos, usan frecuentemente HMI para controlar máquinas, platas o edificios.

La sofisticación de la HMI varía en función de la complejidad de la máquina o sistema para el que se usa. También varía según el uso de la HMI, ya sea para supervisar una máquina o para otros fines, como supervisar operaciones de una planta y controlar equipos.

Cuando el sistema de supervisión, control y adquisición de datos (SCADA) se comunica con los controladores lógicos programables (PLC) y los sensores de entrada/salida para obtener información sobre el funcionamiento de los equipos, esa información se muestra en una HMI, ya sea en un gráfico, gráfica u otra representación visual que sea fácil de leer y entender. Con una HMI, podrá ver toda la información de rendimiento de los equipos de una instalación en un solo lugar, mejorando la visibilidad de las operaciones de la planta. Los operarios también pueden ver y gestionar las alarmas con una HMI asegurando que se puedan tratar rápidamente.

Los operarios también pueden controlar los equipos con las HMI para aumentar la productividad o adaptarse a las circunstancias cambiantes. Pueden hacer ajustes basados en los datos que ven en la HMI, haciendo que este proceso sea más rápido y sencillo.

A medida que el Internet de las Cosas (IoT) sigue desempeñando un importante papel en las instalaciones industriales, las HMI se vuelven más útiles. Pueden usarlas para visualizar datos y controlar los distintos equipos conectados de sus instalaciones.

#### **9.4.1.2. HMI y SCADA**

HMI y SCADA se confunden por sus similitudes y por que trabajan juntos. La mayoría de veces la HMI forma parte de un sistema SCADA.

Un sistema SCADA se utiliza para controlar sistemas grandes, como una planta entera. Es una combinación de otros sistemas como PLCs, sensores y unidades de terminales remotas (RTU). Un sistema SCADA recopila y registra datos, y puede controlar las operaciones automáticamente.

Una HMI, por su parte, es una interfaz gráfica para interactuar con un sistema SCADA y otros sistemas y equipos. Ambos elementos esenciales de un sistema de control industrial más grande. Mientras que el SCADA recopila y almacena los datos, la HMI permite a los usuarios interactuar con los equipos y gestionarlos mediante un panel. Ambos son necesarios. Sin SCADA, una HMI no tendría información que mostrar ni podría controlar los equipos. Sin una HMI, los usuarios no podrían ver los datos que el sistema SCADA recopila ni decirle cómo controlar los equipos.

SCADA y HMI son partes de un sistema más grande. SCADA funciona en segundo plano mientras que la HMI es normalmente el único elemento con el que interactúan los usuarios. Por ello a veces se hace referencia a ellos conjuntamente.

#### **9.4.1.3. Ventajas de la HMI**

Las HMI proporcionan una serie de ventajas en las organizaciones industriales tales como:

**Mayor Visibilidad:** una HMI de alto rendimiento le ofrece una mayor visibilidad de sus operaciones en todo momento, lo que le permite ver el rendimiento de sus equipos o instalaciones desde un solo panel, al que puede acceder incluso remotamente, aumentando la productividad con el tiempo y respondiendo a las alertas con mayor rapidez.

**Eficiencia Mejorada:** gracias al acceso en tiempo real de la HMI a los datos, se puede usar para supervisar la producción y ajustarla a los requisitos cambiantes. La visualización de los datos, combinados con las tecnologías de análisis de datos, le permiten identificar áreas en las que puede mejorar la eficiencia de sus operaciones.

**Menos períodos de inactividad:** con las alertas en un panel central puede responder a los problemas con más rapidez, reduciendo períodos de inactividad. Visualizar y analizar los datos de

rendimiento de los equipos también le pueden ayudar a detectar futuros problemas mecánicos y atajarlos antes de que puedan provocar períodos de inactividad más significativos.

**Usabilidad Mejorada:** las HMI facilitan a los usuarios ver y entender los datos y equipos de control. Presentan los datos en forma de gráficos, gráficas, etc. Permitiendo a los usuarios interpretarlos rápidamente.

**Sistema Unificado:** con las HMI puede controlar todos los equipos usando la misma plataforma, lo que facilita a los operarios su control. También puede ver todos los datos en una ubicación, para tener una visión general clara de las instalaciones. Además, todos los usuarios reciben actualizaciones en tiempo real, con lo que su equipo estará siempre en la misma página.

#### 9.4.1.4. El futuro de la HMI

A medida que los datos sumen un papel más importante en los procesos industriales, la HMI está llamada a ser aún más importante. La tecnología ha avanzado significativamente en los últimos años, pero seguirá evolucionando en el futuro.

Más empresas están cambiando a las HMI de alto rendimiento, que atraen la atención de los usuarios solo a la información más crítica, ayudándoles a dar sentido a todos los datos disponibles y evitando el exceso de información. Las empresas también usan cada vez más pantallas multitáctiles, supervisión remota y sistemas en la nube.

A medida que la tecnología evolucione, las empresas utilizarán más análisis de datos avanzados e inteligencia artificial para obtener más conocimientos, comunicándose a los usuarios a través de las HMI que, en el futuro, también incorporarán realidad aumentada (AR), superponiendo gráficos digitales sobre el mundo real, y tecnología de realidad virtual (VR), que sumerge a los usuarios a un mundo digital para proporcionarles información visual más efectiva. Como la automatización sigue desempeñando un papel más central en los procesos industriales, los usuarios pueden aprovechar las HMI para supervisar y ajustar las actividades automatizadas.



Figura 19: Ilustración de una HMI

## 9.5. Servomotores Industriales

Un servomotor es un actuador rotativo o motor que permite un control preciso en términos de posición angular, aceleración y velocidad, capacidades que un motor normal no tiene. Utiliza un motor normal y lo combina con un sensor para la retroalimentación de posición.

El controlador es la parte más sofisticada del servomotor, ya que está diseñado específicamente para este fin.

### 9.5.0.5. Usos del servomotor

Los servomotores no son en realidad una clase específica de motor, si no una combinación de piezas específicas, que incluyen un motor de corriente continua o alterna, y son adecuados para su uso en un sistema de control de bucle cerrado.

Se utilizan básicamente en la robótica industrial, en la fabricación con sistemas de automatización y en aplicaciones de mecanizado de control numérico (CNC) por ordenador.

El servomotor es un servomecanismo de bucle cerrado que utiliza la retroalimentación de posición para controlar su velocidad de rotación y posición. La señal de control es la entrada, ya sea analógica o digital, que representa el comando de posición final para el eje.

El codificador o encoder sirve como sensor, proporcionando retroalimentación de velocidad y posición. En la mayoría de los casos, sólo se informa de la posición. La posición final se informa al controlador y se compara con la entrada de posición inicial, y luego, si hay una discrepancia, se mueve el motor para llegar a la posición correcta.

Los servomotores más sencillos utilizan motores de corriente continua y detección de posiciones a través de un potenciómetro y también utilizan un control de gran potencia, lo que significa que el motor se mueve a la velocidad máxima hasta que se detiene en la posición designada.

Los sofisticados servomotores para uso industrial disponen de sensores de posición y velocidad, así como de algoritmos de control proporcional-integral-derivativo, lo que permite llevar el motor a su posición de forma rápida.

### 9.5.0.6. Funcionamiento

Los servomotores se controlan enviando un pulso eléctrico de ancho variable, o modulación de ancho de pulso (PWM), a través del cable de control. Hay un pulso mínimo, un pulso máximo y una frecuencia de repetición.

Por lo general, un servomotor sólo puede girar  $90^\circ$  en cualquier dirección para un movimiento total de  $180^\circ$ . La posición neutra del motor se define como la posición en la que el servomotor tiene la misma cantidad de rotación potencial tanto en el sentido de las agujas del reloj como en el sentido contrario.

El PWM enviado al motor determina la posición del eje, y se basa en la duración del pulso enviado a través del cable de control; el rotor girará a la posición deseada.

El servomotor espera ver un pulso cada 20 mS, y la longitud del pulso determinará hasta dónde gira el motor. Por ejemplo, un pulso de 1.5 mS hará que el motor gire a la posición de 90°.

Si el tiempo es inferior a 1.5 mS, se mueve en sentido contrario a las agujas del reloj hacia la posición de 0°, y si el tiempo es superior a 1.5 mS, el servo girará en sentido de las agujas de reloj hacia la posición de 180°.

Cuando se les ordena a los servos que se muevan, estos se moverán a la posición y mantendrán esa posición. Si una fuerza externa empuja contra el servomotor mientras el servomotor mantiene una posición, el servo se resistirá a salir de esa posición.

La cantidad máxima de fuerza que puede ejercer el servo se denomina para de torsión del servo. Sin embargo, los servos no mantendrán su posición para siempre; el pulso de posición debe repetirse para indicar al servo que se mantenga en posición.

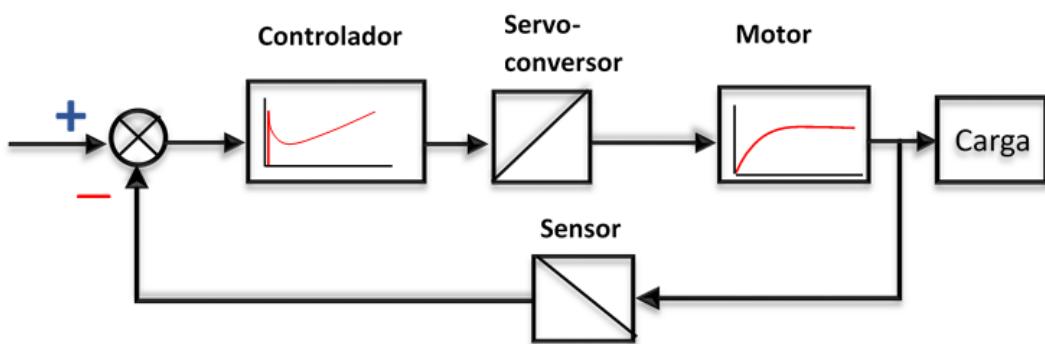


Figura 20: Control de lazo cerrado de un servomotor

#### 9.5.0.7. Componentes de un servomotor

Los servos incluyen tres componentes principales: un motor, un variador/amplificador y un mecanismo de retroalimentación. También se incluye típicamente una fuente de alimentación y un servocontrolador capaz de controlar un solo eje o coordinar el movimiento a varios ejes.

Los servomotores pueden ser del tipo CA o CC, siendo los servomotores de CA los más adecuados para aplicaciones de velocidad constante y los servomotores CC para aplicaciones de velocidad variable.

La retroalimentación es proporcionada normalmente por un codificador o encoder (ya sea interno o externo). En aplicaciones que requieren un posicionamiento muy preciso, se pueden utilizar dos dispositivos de retroalimentación: Uno en el motor para verificar el rendimiento del motor y otro en la carga para verificar la posición real de la carga.

Un servo-accionamiento amplifica la señal de un controlador maestro proporcionando suficiente corriente(potencia) al motor para generar velocidad y producir par. En un motor rotativo, la corriente es proporcional al par, por lo que el servomotor controla directamente el par producido por el motor.

Del mismo modo, en un motor lineal, la corriente es proporcional a la fuerza, por lo que el accionamiento controla la fuerza producida por el motor.

El servomocontrolador (también conocido como controlador de movimiento) puede ser considerado como el cerebro del sistema del servomotor. Aquí es donde reside el perfil de movimiento, incluyendo aceleración, velocidad y desaceleración deseadas.

El controlador envía señales al convertidor, lo que hace que el motor ejecute el movimiento deseado. También tiene la importante tarea de cerrar el bucle en el sistema leyendo la retroalimentación del encoder y modificando la señal del motor (a través del convertidor) para corregir cualquier error en la posición real frente a la deseada, velocidad o par.

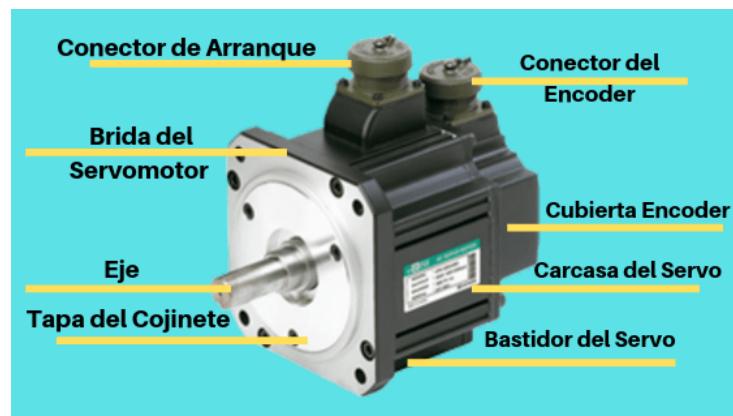


Figura 21: Partes externas del servomotor

#### 9.5.0.8. Ventajas y desventajas de servomotores

##### Ventajas:

- Si se coloca una carga pesada en el motor, el servo-controlador aumentará la corriente en la bobina del motor mientras intenta girar el motor. Básicamente, no hay ninguna condición fuera de los pasos.
- El funcionamiento a alta velocidad es posible.

## Desventajas:

- Dado que el servomotor trata de girar de acuerdo a los pulsos de mando, pero se queda atrás, no es adecuado para el control de precisión de la rotación.
- Mayor costo.
- Cuando está parado, el rotor del motor continúa moviéndose hacia adelante y hacia atrás con un pulso, por lo que no es adecuado si necesita evitar la vibración.

### 9.5.0.9. Aplicaciones

Los servomotores se utilizan en aplicaciones que requieren variaciones rápidas de velocidad sin sobrecalentar el motor.

Algunas de sus aplicaciones son:

- En las industrias se utilizan máquinas herramienta, embalaje, automatización de fábricas, manipulación de materiales, conversión de impresión, líneas de ensamblaje y muchas otras aplicaciones exigentes; robótica, maquinaria CNC o fabricación automatizada.
- También se utilizan en aviones radio-controlados para controlar la posición y el movimiento de los ascensores.
- Se utilizan en robots debido a su suave encendido y apagado y a su preciso posicionamiento.
- También se utilizan en la industria aeroespacial para mantener el fluido hidráulico en sistemas hidráulicos.
- Se utilizan en juguetes controlados por radio.
- Se utilizan en dispositivos electrónicos tales como DVDs, reproductores de discos o impresoras.
- También se utilizan en automóviles para mantener la velocidad de los vehículos.



Figura 22: Servomotores industriales

## 10. Desarrollo de actividades

Como en todo inicio de proyecto, se debe de comenzar con la definición de tareas y actividades que el proyecto tendrá para alcanzar el objetivo final. Esta planificación de actividades se debe de definir con una aproximación del tiempo en que se necesita llevar a cabo cada una de las actividades.

Siguiendo con esta metodología, a continuación se presenta el cronograma general de actividades para llevar a cabo el proyecto de residencia.

### 10.1. Cronograma de Actividades

Actividades	Marzo	Abril	Mayo	Junio	Julio
Capacitación en uso de servomotor, sensores y actuadores del banco de pruebas					
Capacitación interfaz de programación gráfica					
Implementación de comandos de control de servomotor de manera manual					
Implementación de comandos de control de servomotor de manera automatizada					
Integración de la HMI completa					
Redacción de informe final					
Entregable: Informe técnico por escrito					

Figura 23: Cronograma general del proyecto de residencia

El anterior cronograma fue el cronograma preliminar presentado al inicio del proyecto, en esencia general son las tareas que se llevaron a cabo a lo largo del desarrollo, algunas tareas se integraron al cronograma, otras tareas se modificaron, pero en relación al tiempo de ejecución de cada una, se siguió de forma correcta.

## 10.2. Definición de proyecto

El proyecto consiste en desarrollar un control y monitoreo de un banco de pruebas que CIA-TEQ Zapopan comenzó a desarrollar, pero que se encuentra semi-automatizado. Es por ello que el trabajo de este proyecto de residencia es llevar a cabo la integración de los dispositivos actuales a una sola plataforma de control, y así dar inicio a la automatización del banco de pruebas a través de una aplicación de escritorio o HMI, donde se pueda generar pruebas a la máquina eléctrica desde la aplicación.

El banco de pruebas debe de funcionar en dos modos de operación, uno en modo motor y otro en modo generador.

### 10.2.1. Modo Motor

Lo que se pretende realizar en el modo motor es utilizar el banco de pruebas como un monitor de comportamiento de un motor externo, conectado al banco de pruebas.

El siguiente diagrama a bloques muestra como los dispositivos del banco de pruebas están organizados:

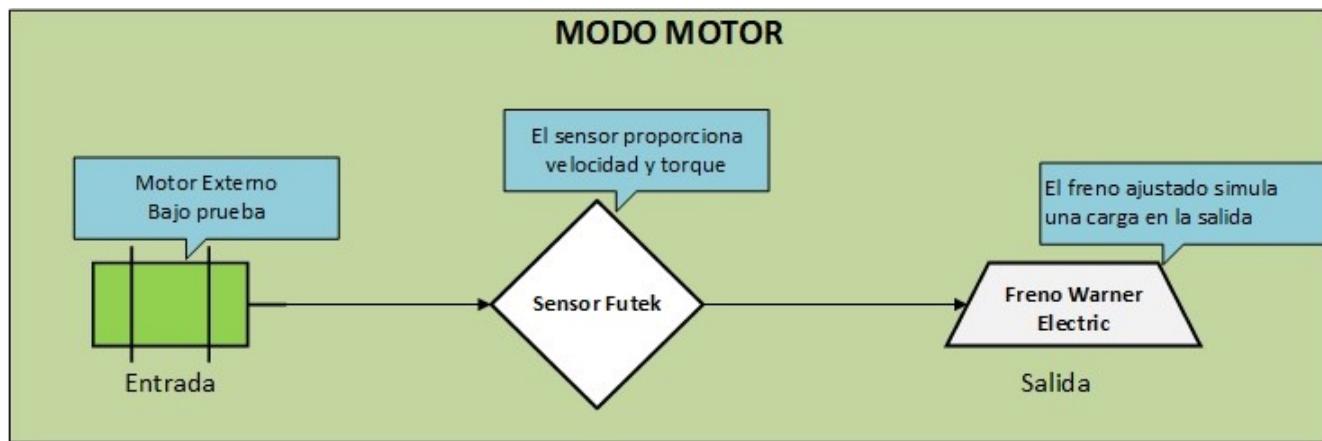


Figura 24: Diagrama a bloques modo motor

Como se observa, en el modo motor se coloca un motor en la entrada del sistema (ilustrado en color verde), este motor es el que se va a poner bajo prueba de funcionamiento.

El siguiente elemento es el sensor Futek, este sensor nos va a proporcionar información de velocidad y torque que el motor esta ejerciendo.

Por último se tiene la salida, el cual para fines de simulación de una carga colocada en el motor, se utiliza un freno Warner Electric en modalidad de freno ajustado, para simular una carga.

### 10.2.2. Modo Generador

En el modo generador, el banco de pruebas se utiliza para poner bajo prueba un generador eléctrico que se encuentra en la salida del sistema.

El siguiente diagrama a bloques muestra como los dispositivos del banco de pruebas están organizados:

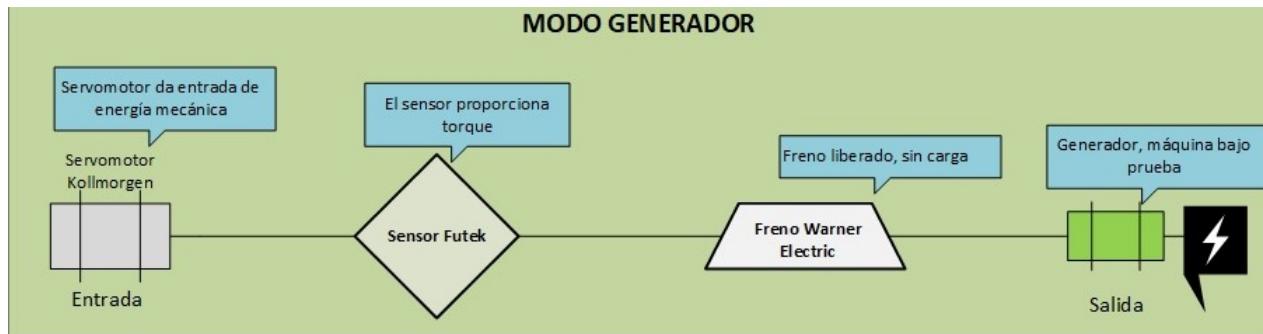


Figura 25: Diagrama a bloques modo generador

Como se puede observar en el diagrama de bloques, en el modo generador se tiene en la entrada del sistema un servomotor Kollmorgen, este servomotor simulará la energía mecánica que entra al eje de distribución, el segundo elemento encontrado es el sensor Futek, que únicamente nos proporciona la información del torque mecánico.

El tercer elemento encontrado en el sistema es el freno Warner Electric, este freno debe de estar en modalidad libre para no añadir una carga extra al sistema.

Por último, se tiene la salida, donde se tiene el generador eléctrico bajo prueba, generando la diferencia de potencial en sus terminales.

Con base a lo anterior, se tiene una mejor idea de como el banco de pruebas debe de trabajar, y son escenarios de trabajo que se deben de tener en cuenta para el desarrollo del control e interfaz gráfica.

### 10.2.3. Primeras ideas

Como primeras ideas, se tienen por lo tanto que el **sistema de control** debe de ser capaz de soportar cambios de modalidad, por lo cual se debe de tener en cuenta que dentro del código del programa de control deben de existir dos bloques, uno para el control del modo motor, y otro para el modo generador.

Dentro de cada bloque, se encuentran los recursos en código de control para los dispositivos del banco de pruebas.

Por otro lado, para el desarrollo de la **interfaz gráfica de usuario**, esta debe de permitir cambios de modos de operación, entre modo motor y modo generador, por lo cual se debe de tener una interfaz gráfica con por lo menos 3 ventanas principales:

- Ventana principal
- Ventana modo motor.
- Ventana modo generador.

Se debe de definir que información de interés se deben de mostrar en las ventanas, por lo tanto:

**Ventana principal:** En esta ventana el usuario tendrá que hacer comunicación con el controlador, saber que el controlador esta en comunicación con la GUI, una vez comunicado, se podrá elegir del menú de opciones el modo en el que se desea trabajar, sea modo motor o modo generador.

**Ventana modo motor:** En esta ventana el usuario podrá observar los datos del sensor futek (velocidad y torque) así como también controlar la liberación y activación del freno ajustado.

**Ventana modo generador:** Esta ventana es la ventana más elaborada de la interfaz gráfica, ya que el usuario dentro de esta ventana podrá observar el monitoreo de señales y el ajuste de movimiento.

Con esta sección se aborda la definición del proyecto, que es parte fundamental para cualquier desarrollo de diseños en ingeniería, al tener definido el proyecto, se tiene un panorama más claro de como es que se va a desarrollar el proyecto, de igual manera surgen preguntas tales como:

- ¿Qué dispositivos son los que se tienen en el banco de pruebas?
- ¿Cómo se comunican?
- ¿Cómo controlo y recibo información?
- ¿Qué controlador utilizar para la aplicación?
- ¿En que plataforma desarollar la GUI?

Las anteriores preguntas es el siguiente paso del desarrollo, serán contestadas en las siguientes secciones.

## 10.3. Reconocimiento de dispositivos

El banco de pruebas a máquinas eléctricas que CIATEQ Zapopan comenzó a desarrollar consiste de los siguientes elementos:

- Kollmorgen Servomotor AKM65K-ACCNR-00
- Kollmorgen Servodrive AKD-P01206-NBEC-0000
- Sensor de torque y velocidad FUTEK TRS605
- Freno Warner Electric CBC-550-90

Los anteriores elementos son los que se encuentran ya en el banco de pruebas pero no hay ninguna plataforma general en el cual estos elementos interactúen en conjunto para poder realizar pruebas a máquinas eléctricas.

Para poder continuar con el desarrollo del proyecto, primeramente se debe de realizar una caracterización de que plataforma de control es la más adecuada para la interacción con los anteriores elementos, tomando en cuenta la comunicación para poder leer y escribir información.

### 10.3.1. Kollmorgen Servodrive y Servomotor

El servomotor Kollmorgen AKM65K-ACCNR-00 necesita trabajar en conjunto con el servodrive Kollmorgen AKD-P01206-NBEC-0000, por lo tanto, el servomotor trabajará bajo órdenes del servodrive.

Con lo anteriormente dicho, se observa que nuestro objetivo es el estudiar el servodrive, y no el servomotor, puesto que dominando y controlando al servodrive, podremos hacer que el servomotor trabaje bajo nuestras órdenes, teniendo al servodrive como interprete intermediario en la comunicación.

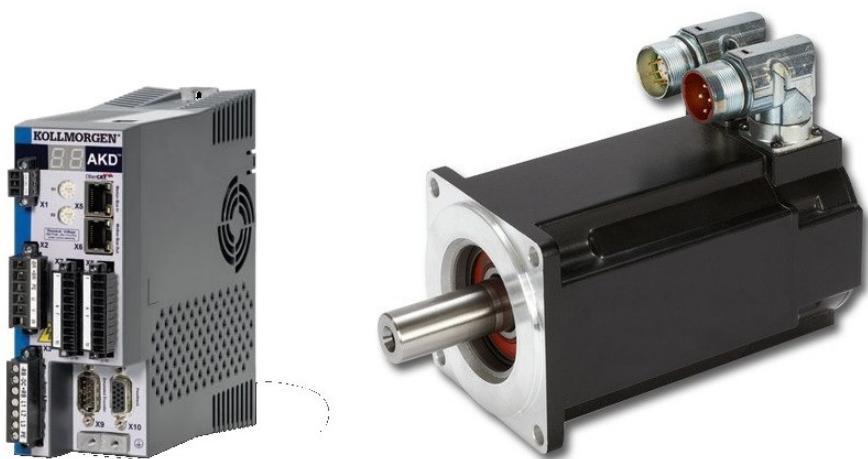


Figura 26: Dispositivos Kollmorgen en banco de pruebas

Nuestro objetivo principal será el conocer el servodrive, por lo tanto se realizó un estudio del manual de usuario propio del servodrive Kollmorgen AKD-P01206-NBEC-0000 para controlar dispositivo servomotor Kollmorgen AKM65K-ACCNR-00.

En primer instancia la manera de poder realizar movimientos con el servomotor, obtener datos propios del servodrive y servomotor es dando uso al software que el fabricante proporciona, este software es llamado "Kollmorgen WorkBench"

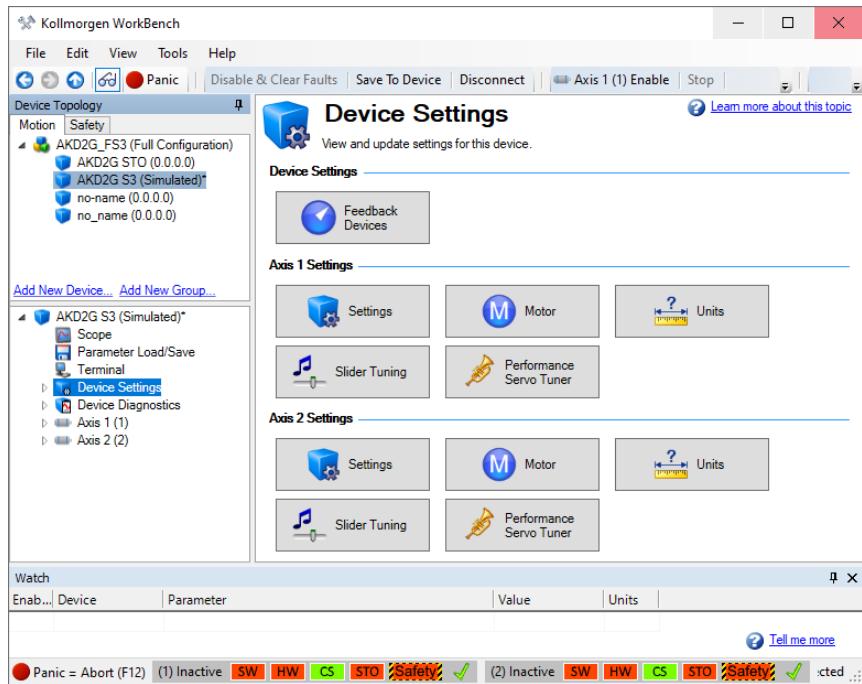


Figura 27: Software Kollmorgen WorkBench

Con ayuda de este software, se pueden realizar movimientos del servomotor con una configuración establecida por el propio usuario.

Este software tiene muchas funciones, muchos datos recopilados, incluso te muestra gráficas de comportamiento y también se pueden crear recetarios de perfiles de movimiento para poder cargarlos directamente.

Este software fue la parte inicial para conocer y aprender como se generan movimientos específicos que el usuario desea realizar.

Para poder dar uso a este software y controlar los dispositivos Kollmorgen solamente se debe de dar uso a una comunicación ethernet de computadora a servodrive, el programa automáticamente reconoce el dispositivo a través de su IP definida y se abre el software para dar inicio a la configuración de perfiles de movimiento.

Una vez que se dominó como configurar diferentes movimientos, se comenzó con la investigación sobre que tipo de comunicación (bus de campo) utilizar para el proyecto.

El servodrive se puede comunicar a través de algún bus de campo tales como:

- EtherCat
- Modbus
- Ethernet TCP /IP

A partir de este punto, se debe de tener en mente que tipo de comunicación debe de soportar el sistema de control para que se pueda comunicar con el servodrive.

Pero este tema se verá más adelante, por el momento, este fue el primer paso para cononocer los dispositivos Kollmorgen que el banco de pruebas tiene y poder ver en funcionamiento el servomotor realizando los movimientos que se establecen previamente en el software Kollmorgen WorkBench.

#### 10.3.2. Sensor de velocidad y torque Futek

El banco de pruebas tiene un sensor de velocidad y torque Futek TRS605. Este sensor no es tan complejo como los dispositivos Kollmorgen que se vieron en el apartado anterior, ya que este sensor entrega señales de salida para el propio controlador.

Este sensor nos entrega el dato del velocidad y torque.



Figura 28: Sensor FUTEK TRS605

### 10.3.2.1. Dato de velocidad

Para obtener el dato de velocidad, se debe de tomar en cuenta las terminales que nos entregan ese dato, dando uso a la hoja técnica del propio sensor:

ANGLE CONNECTOR CODES		
PIN	COLOR	DESCRIPTION
<b>B</b>	Blue	Signal (Angle 1)
<b>E</b>	Black	Ground
<b>G</b>	Brown	Signal (Angle 2)
<b>H</b>	Orange	Power

Figura 29: Terminales involucradas para dato velocidad

Para obtener el dato de velocidad, es necesario conectar del siguiente modo las anteriores terminales:

- **Blue:** Esta terminal es el ángulo 1, debe de ser leída por el controlador.
- **Black:** Esta terminal es ground o por su traducción tierra, conectarla a tierra del sistema.
- **Brown:** Esta terminal es el ángulo 2, debe de ser leída por el controlador.
- **Orange:** Esta terminal es power, debe de ser conectada a +5V

Las terminales de ángulo son las que nos entregan los valores necesarios para determinar la velocidad, esta señal debe de ser leída a través de un controlador que tenga interrupciones, puesto que de esta manera es leída esta señal.

A través de las interrupciones, se establece una temporización entre el ángulo 1 y ángulo 2 para medir su desfase entre una y otra y así se determina la velocidad de giro y la dirección en la que gira el eje.

Como se puede deducir entonces, el ángulo 1 y el ángulo 2 son señales cuadráticas, las cuales son pulsos cuadrados que se modifican de acuerdo a la velocidad y giro del eje, como se puede apreciar en la siguiente imagen:

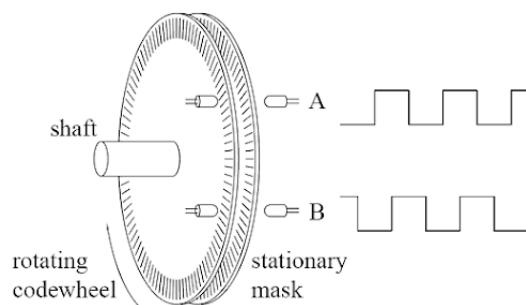


Figura 30: Señales cuadráticas de encoder

#### 10.3.2.2. Dato de torque

Para obtener el dato de torque, es necesario conocer las siguientes terminales:

TORQUE CONNECTIONS		
PIN	COLOR	DESCRIPTION
C	Green	+ Voltage Output
D	White	- Voltage Output
E	Black	Ground
F	Red	Power Supply

Figura 31: Terminales involucradas para dato torque

Para obtener el dato del torque, es necesario conectar del siguiente modo las anteriores terminales:

- **Green:** Esta terminal es la salida de voltaje positivo, debe de ser leída por el controlador.
- **White:** Esta terminal es la salida de voltaje negativo, debe de ser leída por el controlador.
- **Black:** Esta terminal es ground o por su traducción tierra, conectarla a tierra del sistema.
- **Red:** Esta terminal es Power, debe de ser conectada a +12V

Para poder reconocer el dato del torque, se deben de tratar las señales de voltaje positivo y negativo como dos señales analógicas en el sistema, con la observación de que, son señales diferenciales por lo cual se deben de acondicionar sea en el propio sistema de control, o usando algún hardware externo al sistema, para dejar una señal acondicionada para ser leída por el controlador.

#### 10.3.3. Freno industrial Warner Electric

El siguiente dispositivo a reconocer es el freno industrial Warner Electric CBC-550-90. Este dispositivo es un freno industrial el cual cuenta con su controlador propio, pero a diferencia del controlador del servomotor, este puede ser accionado por un controlador master a través de señales digitales.

El propósito de este dispositivo en el banco de pruebas es para generar alguna resistencia o carga en el sistema cuando el banco de pruebas se encuentra operando en el modo motor, y que cuando el banco de pruebas se encuentre trabajando en modo generador, se encuentre libre de carga en el sistema, osea sin freno.

Lo interesante de este dispositivo, es que tiene 3 modalidades de operación:

- Freno total
- Freno Ajustado
- Freno Libre

En **Freno total** como su nombre lo indica, esta totalmente amarrado al sistema, no puede generar ningún movimiento el eje del freno, ya que el freno mecánico ha sido accionado. Este modo para el proyecto no es necesario implementarlo, puesto que significa un peligro, ya que si se llegará a accionar accidentalmente este tipo de modo cuando se encuentra el banco de pruebas ejecutando una prueba, puede correr un riesgo significativo para el banco, por lo cual se debe de implementar una solución para prevenir este modo de operación de freno.

En **Freno ajustado**, este modo de operación permite accionar un tipo de freno ajustado, el cual este modo de operación será utilizado como una simulación de una carga extra en el sistema cuando se encuentra trabajando en modo motor. El ajuste se realiza a través de un potenciómetro embebido que se tiene en la tarjeta de control/potencia del freno, y el ajuste se debe de realizar de manera manual.

En **Freno libre**, en este modo el freno se encuentra libre, no debe de añadir carga extra al sistema, por lo cual este modo de operación de freno será utilizada cuando se encuentre el banco de pruebas en modo generador.

De manera física, los dispositivos son los siguientes:

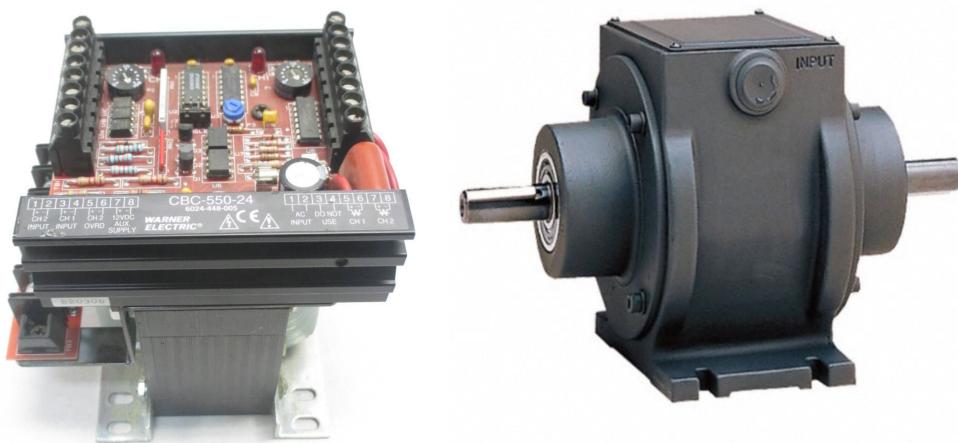


Figura 32: Controlador y freno industrial Warner Electric

Para poder controlar el freno industrial se debe de seguir la siguiente tabla:

## Mode Options

### Single Mode

Ch1	Ch2	Ch2 Ove	Ch1 output
0	0	0	No change
0	0	1	On at full torque
0	1	0	Off
0	1	1	On at full torque
1	0	0	On at adjusted torque
1	0	1	Full torque
1	1	0	On at adjusted torque
1	1	1	Full torque

Figura 33: Tabla lógica de control Freno Warner Electric

La anterior tabla lógica se encuentra marcada con colores de los modos de operación que son de interés en el proyecto, el cual a continuación se describen cada uno:

- **Azul:** El color azul será utilizado como "No change", se mantendrá en un modo comportándose como latch.
- **Amarillo:** En el color amarillo se encuentra el modo generador, en otras palabras el freno se encuentra libre.
- **Café:** En el color café se encuentra el modo motor, el freno se encuentra en un torque ajustado.
- **Rojo:** En el color rojo se encuentra el estado donde el freno mecánico se encuentra presente, estos estados se deben de evitar.

Al analizar la tabla anterior se puede observar que en los estados lógicos que significan un peligro para el banco de pruebas (los marcados en rojo) hay una entrada denominada CH2 OVE que se encuentra en '1' lógico siempre, y que en los tres estados lógicos que vamos a utilizar para el proyecto, esa entrada se mantiene en '0' lógico, por lo cual una forma garantizada de que nunca vamos a entrar en el estado lógico de color rojo es aterrizando a '0' lógico la entrada CH2 OVE y así vamos a evitar ese estado lógico no deseado en el sistema.

Por último, observamos entonces que para poder controlar el freno adecuadamente para el sistema, será necesario dar uso a las entradas CH1 y CH2 siguiendo los estados lógicos marcados en azul, amarillo y café.

Como se observó en esta sección, se dio inicio al reconocimiento de los dispositivos encontrados en el banco de pruebas y ya tenemos un bosquejo de que tipo de controlador se debe de utilizar.

El sistema de control que se debe de utilizar entonces, debe de soportar lo siguiente:

- Comunicación industrial para servodrive (EtherCat,Modbus,Ethernet TCP IP).
- Pines de entrada con interrupción (x2) para Sensor Futek.
- Pines de entrada analógica diferencial (x2) para Sensor Futek.
- Pines de salida digital (x2) para freno industrial.

Por lo tanto, este dato es lo mínimo que se requiere para el sistema de control, que será analizado en la siguiente sección.

## 10.4. Propuestas de controlador

Para la elección del controlador del proyecto, se tienen los siguientes dispositivos posibles a utilizar:

- *Raspberry Pi 3B*
- *Arduino UNO*

### 10.4.1. Raspberry Pi

La raspberry Pi como definición es una mini computadora embebida con un procesador Broadcom BCM2837B0.

Este sistema embebido es muy utilizado en proyectos orientados a la mecatrónica e informática, lo que hace interesante una raspberry es que cuenta con un procesador, el cual a este se le carga una imagen de algún sistema operativo, sea Windows, o alguna distribución de Linux. Logrando así tener un recurso adicional para proyectos orientados a la mecatrónica.

Teniendo el recurso de tener un sistema operativo en la propia tarjeta embebida, se pueden desarrollar proyectos mecatrónicos más robustos que cuando se usa un microcontrolador, puesto que con el sistema operativo o mejor dicho con ayuda del procesador, se pueden crear proyectos donde se den uso a los pines GPIO de la raspberry, se tome lectura de sus entradas y salidas, y que además los datos sea almacenados en una nube directamente.

Teniendo un sistema operativo en el proyecto, se abren muchas posibilidades de continuar con el desarrollo, incluso involucrando tecnologías IoT.

Para el proyecto de residencia se da como propuesta usar Raspberry Pi como controlador del proyecto, ya que la Raspberry tiene la capacidad de soportar comunicación EtherCat y Modbus, ya que la tarjeta tiene un slot Ethernet embebida.

Se pretendía usar una plataforma llamada Codesys el cual es una plataforma donde la Raspberry la podemos controlar como si fuera un PLC.

Al utilizar Codesys, se podría cargar la configuración propia de una comunicación Modbus o EtherCat que el servodrive Kollmorgen soporta y así poder comunicarnos con la tarjeta de control.

Posterior a ello, la configuración de los espacios de memoria (registros de configuración) Kollmorgen pudieran haber sido de manera más rápida, ya que se encontró el archivo de configuración GSD del servodrive para cargarlo a Codesys.

Además otra ventaja de usar Raspberry sería su capacidad de correr multitareas o como se le conoce en informática "multithreading" para tener la ventaja de correr dos o tres tareas al mismo tiempo sin afectar el proceso de ejecución del programa.

Pero una desventaja que se observó con esta propuesta es que para utilizar algunas funciones requeridas para el proyecto por parte de Codesys, entraba en juego una licencia de pago para poder usar esas características, por lo cual la propuesta quedó por un lado.

Pero sin duda alguna, trabajar con Raspberry Pi bajo la plataforma de Codesys se pudieran desarrollar proyectos muy interesantes.

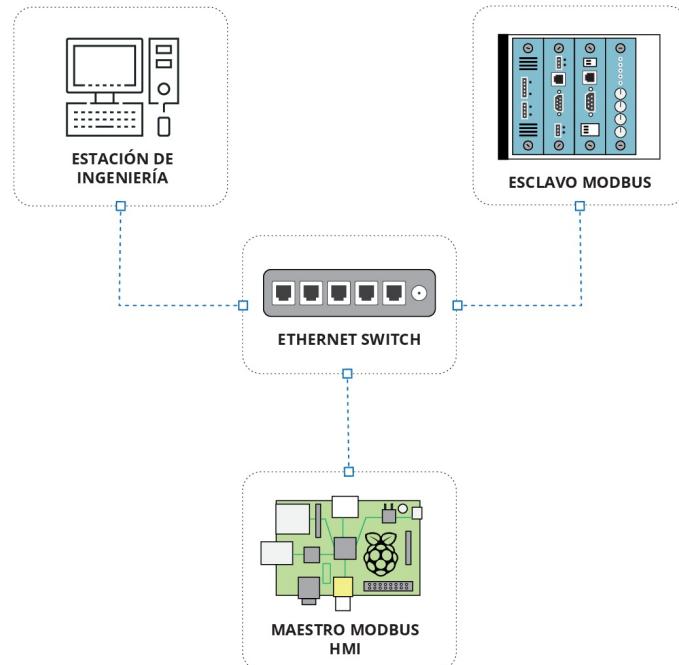


Figura 34: Red de comunicación Raspberry Pi

#### 10.4.2. Arduino

El arduino Uno como se vio en la parte de teoría, es un sistema embebido el cual cuenta con un microcontrolador Atmega328p AVR de 8 bits, el arduino funciona de manera excelente para proyectos de pequeña escala el cual el fabricante indica que es una tarjeta de enseñanza para la inicialización al mundo de la programación de microcontroladores.

En la mayoría de universidades, el arduino forma parte de la enseñanza en asignaturas como "programación de microcontroladores", "programación de sistemas embebidos", entre otros.

Para el proyecto de residencia se toma en cuenta la capacidad del arduino uno para utilizarlo como controlador general del banco de pruebas a máquinas eléctricas, únicamente se debe de contemplar que cumpla con los requisitos mínimos que los dispositivos del banco de pruebas requieren.

Por lo tanto se comenzó con la investigación de como realizar una comunicación EtherCat o Modbus para poder enlazar la comunicación de Arduino a Servodrive bajo alguno de estos dos protocolos de comunicación.

Se comenzó con la investigación de EtherCat, si era posible realizar ese tipo de protocolo de comunicación en arduino Uno, el cual la información que se encontró no estaba del todo clara de como realizar una comunicación bajo este protocolo, por lo cual se decide por investigar con más profundidad la comunicación Modbus TCP/IP en arduino.

En la etapa de investigación se encontró un curso que ayudó bastante para observar que el arduino puede soportar comunicación Modbus TCP/IP, este curso se encuentra en la página Udemy.com, el nombre del curso es *"How to program an Arduino as a Modbus TCP/IP Client and Server"* curso realizado por el instructor Emile Ackbarali.

Con el anterior curso, logré realizar pruebas de comunicación Modbus TCP/IP entre Arduino UNO y PC, usando un simulador de interpréte Modbus, pude observar la comunicación y la manera de interactuar escribiendo y leyendo registros de modbus a través de arduino, siendo este un cliente en la comunicación y la computadora un servidor.

Con base a la experiencia anterior, se encuentra una manera eficiente para establecer la comunicación al servodrive Kollmorgen, únicamente se tendría que revisar los registros involucrados para la configuración correcta del servodrive y hacer pruebas de lectura/escritura, esta etapa se verá más adelante, por el momento se llega a una solución para establecer la comunicación Modbus TCP/IP desde el arduino funcionando como cliente.

Cabe mencionar que el arduino no tiene slot ethernet, por lo tanto para realizar esta comunicación se dió uso al *ethernet shield for arduino* para darle la capacidad al arduino.

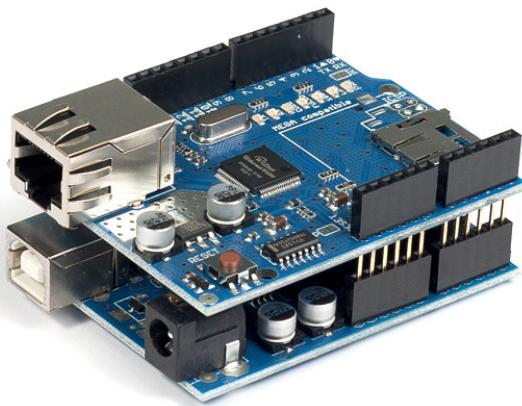


Figura 35: Arduino Uno con Arduino Ethernet Shield

Además del shield, se dieron uso a las siguientes librerías para realizar la comunicación Modbus, las cuales fueron:

- *ModbusTCP library for the Arduino*
- *Ethernet library*

Las anteriores librerías son de uso libre para proyectos.

Una vez concluido de que el arduino puede mantener una comunicación modbus, se comienza a revisar los demás requisitos para los otros dispositivos del banco de pruebas.

Para poder controlar el freno industrial Warner Electric, se necesitan salidas digitales del controlador, cosa que el arduino tiene.

Para el sensor Futek TRS605, para leer el dato de velocidad se necesita pines de interrupción, cosa que el propio microcontrolador del arduino soporta.

Por el lado del dato del torque, es una señal analógica por lo cual el Arduino tiene pines dedicados para señales analógicas.

Al final es más que evidente que el arduino ha cumplido con los requisitos para ser el controlador del proyecto de residencia y poder controlar y leer los dispositivos del banco de pruebas.

Por último, se revisan las capacidades de memoria de las diferentes versiones de tarjetas arduino:

Arduino	Processor	Flash	SRAM	EEPROM
UNO, Uno Ethernet, Menta, Boarduino	Atmega328	32K	2K	1K
Leonardo, Micro, Flora, 32U4 Breakout, Teensy, Esplora	Atmega 32U4	32K	2.5K	1K
Mega, MegaADK	Atmega2560	256K	8K	4K

Figura 36: Comparativa de capacidad de memoria arduino

Con base a la anterior comparativa, se observa una significante limitación en la memoria del arduino Uno, tanto en la memoria Flash como en la memoria RAM.

Como recordatorio, la memoria Flash es donde se almacena el programa (código) del control, y la memoria RAM es la memoria donde se almacenan datos/variables al ejecutar el programa de control.

Con la limitada capacidad de memoria del Arduino Uno, se opta por elegir el Arduino Mega como el controlador general del proyecto, dando así la oportunidad de que el microcontrolador Atmega2560 sea el encargado de llevar el control del proceso.

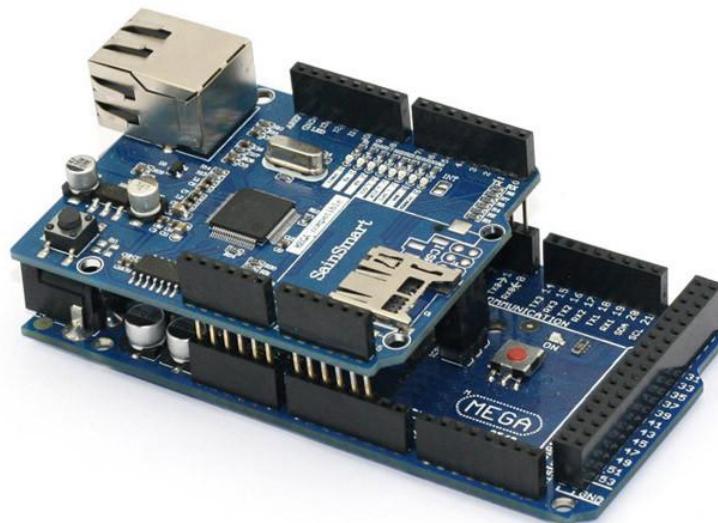


Figura 37: Arduino Mega junto con Ethernet shield for arduino

Al revisar la documentación técnica del encoder Futek TRS605 se observa que para la obtención del dato del torque nos la proporciona en una señal analógica diferencial, esto quiere decir que nos proporciona dos señales analógicas donde una tiene un voltaje positivo y otra tiene un voltaje negativo.

Esto representa un problema para el microcontrolador de Arduino Mega, en este caso el Atmega2560 ya que sus pines analógicos que tiene son referenciados a tierra, por lo tanto debe de ser un voltaje positivo, y no un voltaje negativo.

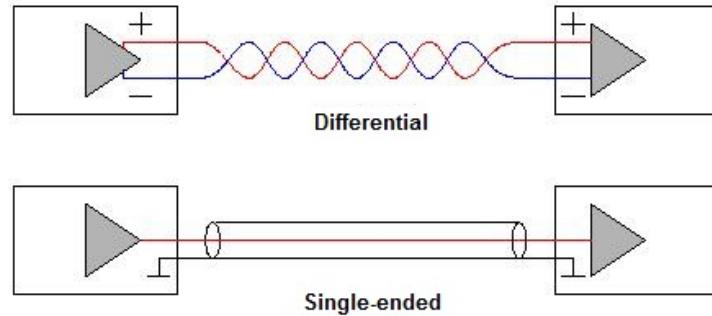


Figura 38: Señal diferencial y señal single ended

Por lo tanto se comienza con la investigación de como acondicionar la señal del sensor Futek para ser leída por el microcontrolador, se llega a la solución de dar uso al ADC ADS1115 ya que este ADC cuenta con entradas diferenciales, que pueden leer dichos voltajes del sensor y acondicionarla internamente, una vez que se acondicione la señal, este dato de señal puede ser leído a través de la comunicación I2C, haciendo la comunicación Arduino - ADS1115.

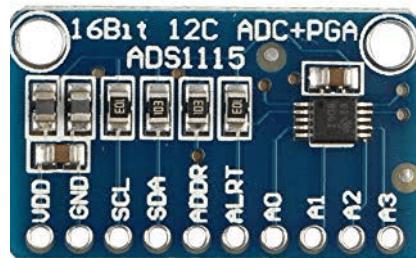


Figura 39: ADC ADS1115

## 10.5. Propuesta para GUI

Para el desarrollo de la interfaz gráfica de usuario (GUI) al inicio de la definición del proyecto se estaba contemplando lo siguiente:

- *MyOpenLab*
- *Visual Studio .Net with CSharp*

MyOpenLab fue la primer propuesta para la GUI, ya que esta plataforma es libre y es muy utilizada en el campo de la enseñanza de interfaces gráficas.

MyOpenLab según la documentación es una herramienta para la programación visual, se usan bloques gráficos para ir desarrollando una aplicación gráfica para el proyecto, algo parecido a MyOpenLab sería Labview, recordando que este último es de pago y esta orientado al desarrollo de VI (Virtual instrument).

El backend de esta plataforma se encuentra desarrollado en Java.

Personalmente se comenzó con la investigación de que tan viable sería dar uso a esta plataforma para la GUI del proyecto de residencia, el cual se encontró con algunas fallas a la hora de ejecutar pruebas sencillas con la comunicación arduino, y que además la comunidad de apoyo por parte de otros desarrolladores no se encontraba activa para las fechas del desarrollo del proyecto, por lo cual fueron puntos a tomar en cuenta. También se observó de que si existiera algún problema con algún bloque de función, se debería de depurar directamente en la parte del desarrollo Java para poder resolver el conflicto.

MyOpenLab puede ser una excelente herramienta para la inicialización del aprendizaje de programación gráfica, para poder crear prácticas o algunos proyectos de pequeña escala, pero para proyectos más grandes sería prudente contemplar otra plataforma con mayor apoyo de la comunidad de desarrolladores, así como también que sea más estable y robusta en sus bloques de funciones.



Figura 40: Aplicación bajo entorno MyOpenLab

Es por ello que se empezó a investigar otra propuesta para la GUI, en este caso se encontró la plataforma Microsoft Visual Studio, para dar uso al framework Windows .Net y el lenguaje Csharp para la creación de la GUI.



Figura 41: Logo Csharp y Microsoft NET

Con una experiencia previa que tuve con Visual Basic en un lugar donde trabajaba, donde logré desarrollar una aplicación de escritorio para la manipulación de un libro de excel para el control de inventarios y generación de listas de materiales, a través de *Visual Basic for Applications Excel*, por lo tanto, cuando observé como se desarrolla una aplicación de escritorio a través de Csharp y el framework Windows .Net de inmediato recordé el desarrollo de aquella aplicación y lo fácil e interactiva que es desarrollar una aplicación a través de código puro.

Solo que esta vez, sería utilizando una plataforma nueva para mi carrera de estudiante, sería utilizar Visual Studio community, la tecnología framework Windows .Net y el lenguaje de desarrollo C sharp, el cual agrega un pequeño desafío en el desarrollo de la GUI, pero que es alcanzable.

Se dio como propuesta realizar la GUI con las herramientas anteriormente mencionadas y fue aceptada.

La utilidad de usar framework .net es tener la capacidad de desarrollar aplicaciones de escritorio nativas para el sistema operativo Windows, el cual esta tecnología proporciona al desarrollador herramientas propias para la creación de aplicaciones de escritorio Windows.

Esta tecnología no únicamente se utiliza para el desarrollo de aplicaciones de escritorio, si no que también tiene utilidad en el desarrollo de aplicaciones móviles, aplicaciones web, desarrollo de interfaces, etc.

Por lo tanto, utilizar framework .net junto con C sharp desglosa muchas posibilidades, pero sus capacidades de estas herramientas son explotadas en carreras de ingeniería de desarrollo software o informática.

Como se puede deducir, al crear la GUI del proyecto con estas herramientas, se mezclan dos áreas de ingeniería, por una parte la ingeniería en mecatrónica/electrónica y por otra la ingeniería en desarrollo software, quedando así un desarrollo de proyecto que abarca estas áreas.

## 10.6. Diagrama de bloques

Se presenta el diagrama de bloques del proyecto, el cual muestra todos los dispositivos involucrados y la manera en que estos están conectados para formar el sistema de control del banco de pruebas a máquinas eléctricas.

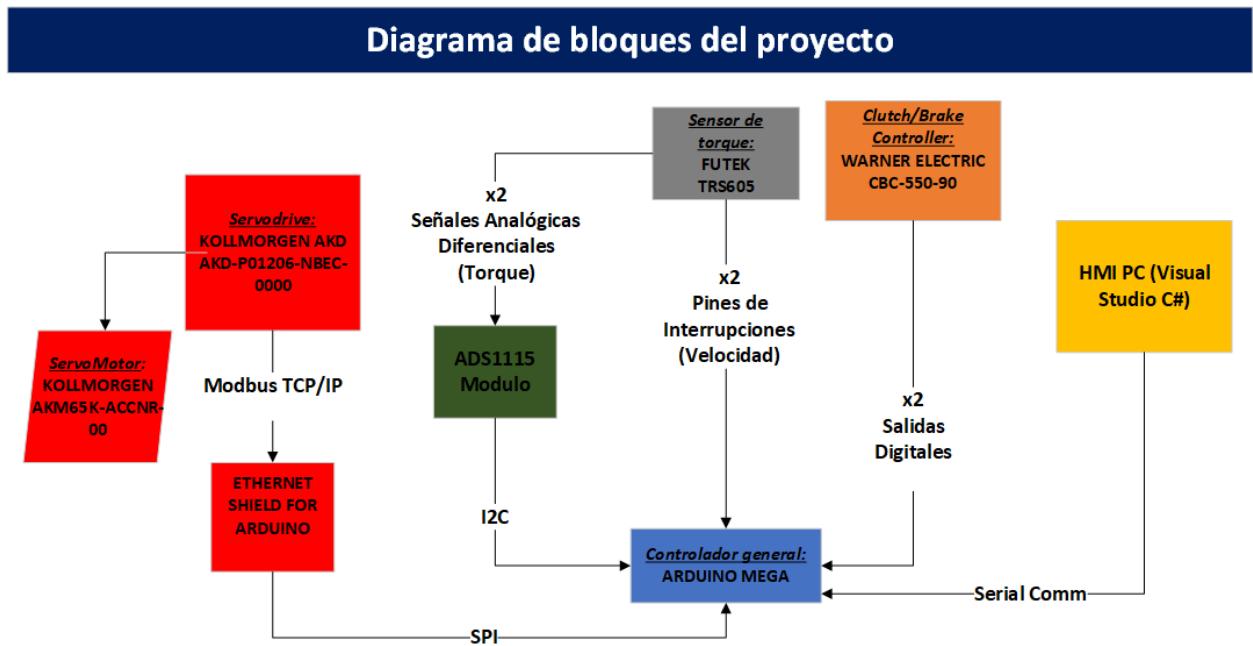


Figura 42: Diagrama de bloques del proyecto

## 10.7. Desarrollo de comunicación Kollmorgen Modbus

En esta sección se presentan las actividades para el desarrollo de la etapa de comunicación hacia el servodrive Kollmorgen AKD.

### 10.7.1. Lectura del manual de usuario

Primeramente, como se definió anteriormente, la comunicación hacia el servodrive Kollmorgen AKD será a través de una comunicación Modbus TCP/IP, realizando así la comunicación entre Arduino Mega y el servodrive, tal como se vio anteriormente en el diagrama de bloques.

La comunicación Ethernet será con ayuda del shield Ethernet for arduino, el cual este módulo tiene como circuito integrado un W5100 que es el controlador que efectua y condiciona la comunicación ethernet modbus TCP/IP.

Pero antes de enlazar la comunicación Modbus Ethernet, se debe de revisar el manual de usuario del servodrive Kollmorgen AKD para conocer los registros de memoria, y comenzar a reconocer cuales de ellos son los que se deben de configurar y leer para poder realizar los movimientos de acuerdo a las condiciones que el usuario establezca a través de la interfaz de usuario.

Se tomó la lectura del manual de usuario del servodrive, sobretodo en la parte donde sale una tabla resumida de los registros de memoria de configuración del servodrive. En dicha tabla se puede observar el nombre del parámetro, tipo y descripción, como se muestra :

Parámetro o comando	Tipo	Descripción
DRV.BLINKDISPLAY (PG 485)	Comando	Hace que la pantalla parpadee durante 10 segundos.
DRV.BOOTTIME (pg 486)	R/O	Muestra la fecha y hora de inicio de la sesión actual.
DRV.CLRFAULTHIST (PG 487)	Comando	Elimina el registro de historial de fallas de la memoria no volátil.
DRV.CLRFAULTS (PG 488)	Comando	Intenta eliminar todas las fallas activas en la unidad.
DRV.CMDDELAY (pg 489)	R/W	Ejecuta un retraso antes de la ejecución del comando siguiente.
DRV.CMDSOURCE (PG 490)	NV	Configura el origen de comando de (servicio, bus de campo, entrada analógica, engranaje, digital o Bode).
DRV.CRASHDUMP (pg 492)	Comando	Recupera información de diagnóstico después de un error en la unidad.
DRV.DBILIMIT (pg 493)	NV	Configura la amplitud máxima de la corriente para freno dinámico.
DRV.DEC (PG 494)	NV	Configura el valor de desaceleración para el bucle de velocidad.
DRV.DIFVAR (pg 496)	R/O	Hace una lista de todos los valores que difieren del valor pre-determinado.
DRV.DIR (pg 497)	R/W	Cambia la dirección de la unidad.
DRV.DIS (PG 499)	Comando	Desactiva el eje (software).
DRV.DISMODE (pg 500)	NV	Selecciona las opciones de comportamiento de desactivación.

Figura 43: Tabla de parámetros resumida

Pero, para poder apreciar en mayor detalle la descripción del parámetro, se debe de ir al número de página que indica entre paréntesis, en esa sección, se explica a detalle todo lo que se debe de saber del parámetro en cuestión:

## 24.13.25 DRV.EN

Información general	
Tipo	Comando
Descripción	Activa el eje (software).
Unidades	N/D
Rango	N/D
Valor pre-determinado	El software de unidad analógico está activado. Todos los demás tipos de software de unidad están desactivados.
Tipo de datos	N/D
Vert también	DRV.DIS (pg 499), DRV.DISSOURCES (pg 502) DRV.ACTIVE (pg 484)
Versión de inicio	M_01-00-00-000

### Variantes admitidas

Variante	Compatible
AKD BASIC	✓
AKDSynqNet	✓
AKD EtherNet/IP	✓

### Información del bus de campo

Bus de campo	Índice/Subíndice	¿Es de 64 bits?	Atributos	¿Tiene signo?	Versión de inicio de objeto
Modbus	254	No	Comando	No	M_01-03-00-000

### Descripción

DRV.EN ejecuta una activación de software a la unidad. Puede consultar el valor de DRV.ACTIVE (pg 484) para comprobar si la unidad está actualmente activada o desactivada. También puede consultar el valor de DRV.DISSOURCES (pg 502) para comprobar si el bit de activación de software es alto (se ejecutó la activación de software mediante la ejecución de DRV.EN) o si el bit de activación de software es bajo (se ejecutó la desactivación de software mediante la ejecución de DRV.DIS). Si el bit de activación de software de la unidad es bajo y se ejecutó DRV.EN, las fallas de la unidad se eliminan automáticamente durante el proceso de activación de software.

Figura 44: Descripción de parámetro detallada

Se debe de observar que en esta sección se encuentra la dirección del bus de campo modbus, el cual esta es la dirección del registro de memoria para poder configurar este parámetro.

### 10.7.2. Teoría básica Modbus TCP/IP

En el conocimiento básico del protocolo Modbus TCP/IP se debe de tener en cuenta en que en la red existen clientes y servidores. Lo anterior es lo mismo que llamar maestro al cliente y esclavo al servidor.

Por lo tanto en cualquier red de Ethernet Modbus TCP/IP , se encontrará el dispositivo que se hace llamar como cliente (que es el que puede leer y escribir) y el servidor que es el que responde bajo los comandos de orden que indica el cliente.



Figura 45: Red Modbus TCP/IP

Como se observa en la figura anterior, en la red de modbus TCP/IP se tienen tanto esclavos (servidor) como maestros (cliente).

Con esta breve explicación se deduce entonces que el arduino debe de trabajar como un cliente y el servodrive debe de trabajar como un servidor, el cual responderá a las ordenes que el arduino envíe.

Como en cualquier protocolo de comunicación, modbus TCP/IP tiene su trama de datos, el cual debe de comprenderse para tener mejor idea de como es que se envían datos a través de esta comunicación:

#### TRAMA MODBUS EN TCP

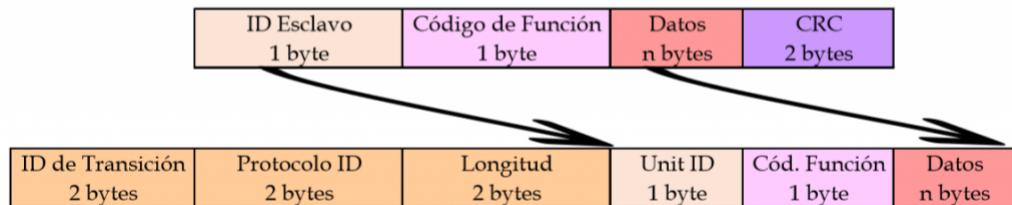


Figura 46: Trama de comunicación modbus TCP/IP

Las partes importantes para la comunicación modbus son los tres últimos bloques de la trama (ID, Código de función, Datos).

El ID por default se deja como 01 (esto ya se encuentra configurado propiamente por la trama).

Se observa también que requiere de un código de función, estos códigos ya se encuentran establecidos propiamente para la comunicación modbus TCP /IP, las cuales se muestran a continuación:

Code	1/16-bit	Description	I/O Range
01	1-bit	Read coils	00001 – 10000
02	1-bit	Read contacts	10001 – 20000
05	1-bit	Write a single coil	00001 – 10000
15	1-bit	Write multiple coils	00001 – 10000
03	16-bit	Read holding registers	40001 – 50000
04	16-bit	Read input registers	30001 – 40000
06	16-bit	Write single register	40001 – 50000
16	16-bit	Write multiple registers	40001 – 50000
22	16-bit	Mask write register	40001 – 50000
23	16-bit	Read/write multiple registers	40001 – 50000
24	16-bit	Read FIFO queue	40001 – 50000

Figura 47: Códigos de funciones para Modbus TCP /IP

La tabla anterior es importante por que es la base del entendimiento de la comunicación modbus.

En el manual de usuario Kollmorgen AKD se encuentra que los registros de memoria propios del servodrive utilizan las funciones:

- 03 - *Read Holding Registers*
- 16 - *Write Multiple Registers*

Con base a la tabla, se observa que ambas funciones que Kollmorgen Servodrive utiliza, son de 16 bits. Esto es importante tomarlo en cuenta, ya que en la descripción detallada de cada parámetro de función del servodrive indica si la palabra es de 8,16,32 o 64 bits. De acuerdo a la longitud de la palabra, será la cantidad de registros que se deben de contemplar para poder igualar a la longitud de la palabra.

En el manual de usuario indica que cuando se accede o escribe en un registro de 32 bits, se deben de contemplar 2 registros (16 bit cada registro) y cuando se accede o escribe en un registro de 64 bits, se deben de contemplar 4 registros.

Con este dato se demuestra que es para que la longitud del dato del registro coincida con la cantidad de registros de tamaño fijo que modbus maneja para dichas funciones.

#### 10.7.3. Pruebas de comunicación Modbus TCP IP

Para tener un mejor entendimiento en la comunicación modbus y observar como es que se leen y escriben en registros, se hicieron varias pruebas de comunicación, a través de la computadora de desarrollo y el propio servodrive kollmorgen.

En primer lugar, se creó una red local, el cual esta misma sería utilizada para el proyecto de residencia. Las IP de los dispositivos son los siguientes:

IP	Device
192.168.0.195	AKD Kollmorgen
192.168.0.192	Arduino Ethernet
192.168.0.194	PC

Figura 48: IPs establecidas para el proyecto

Se procede a configurar la IP local tanto en Windows como en el servodrive. Para poder configurar la IP en el servodrive se dio uso al software Kollmorgen WorkBench.

Por lo tanto, la red quedaría de la siguiente manera:

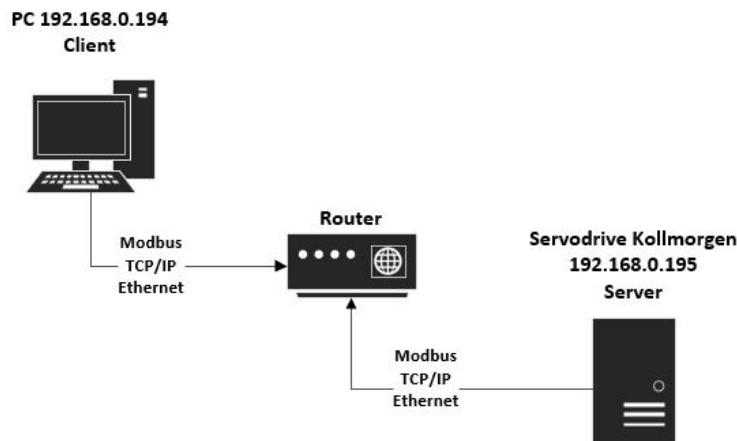


Figura 49: Red local para realizar prueba de comunicación

Una vez realizada la red local para las pruebas de comunicación, se mandó un comando ping a través de la consola de comandos de Windows para observar si hay respuesta por parte del servodrive Kollmorgen a través de red:

```
C:\Users\Miguel Ángel Pérez>ping 192.168.0.195

Pinging 192.168.0.195 with 32 bytes of data:
Reply from 192.168.0.195: bytes=32 time<1ms TTL=255
Reply from 192.168.0.195: bytes=32 time=1ms TTL=255
Reply from 192.168.0.195: bytes=32 time=1ms TTL=255
Reply from 192.168.0.195: bytes=32 time=1ms TTL=255

Ping statistics for 192.168.0.195:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

Figura 50: Respuesta al Ping por parte de servodrive

Como se puede ver, la red ya se encuentra habilitada, por lo tanto se procede a realizar pruebas de escritura y lectura de algunos registros, para ello fue utilizada la herramienta *Modbus poll*.

Con la anterior herramienta podemos escribir o leer registros propios del servodrive, esto se hace enlazando *Modbus poll* a la red local y colocando la IP destino, en este caso la IP del servodrive.

Como ejemplo se realizó la lectura del registro DIN.MODE el cual es la dirección 122 en modbus. Con la herramienta se establece la dirección a leer, la función que va a realizar (este caso 03 Read Holding Register) y la cantidad de registros, en este caso 2 registros, por que el registro DIN.MODE es de 32 bits:

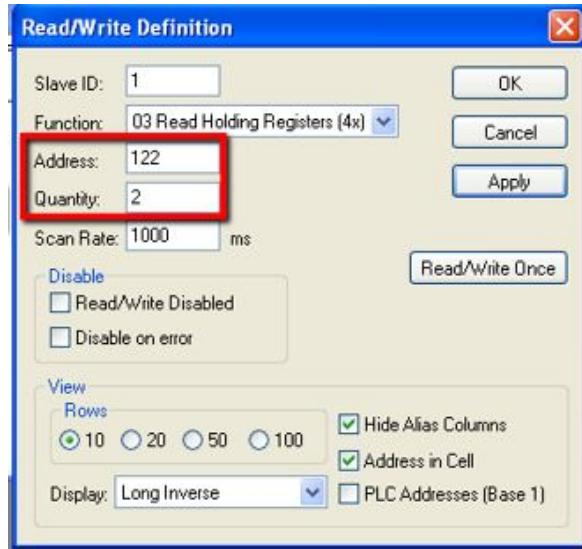


Figura 51: Configuración para leer DIN MODE

Una vez que se configura lo que la herramienta deberá de hacer, se observa en el registro 122 un dato, este dato es el que se encuentra escrito en dicho registro:

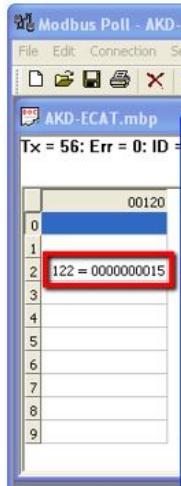


Figura 52: Respuesta al leer registro 122 DIN MODE

Lo interesante de esta herramienta es que se puede observar el flujo de tráfico de la comunicación, observando como se envía el trama de solicitud por parte del cliente y la respuesta por parte del servidor:

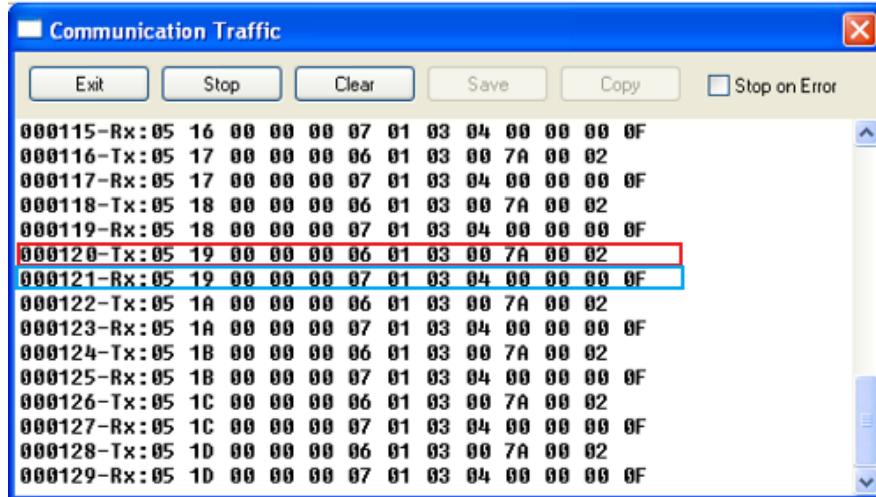


Figura 53: Tráfico de comunicación

Se observa en color rojo, la solicitud enviada por el cliente (TX), el cual la parte importante es donde inicia el 01 03 00 7A 00 02, donde el 01 indica el ID, el 03 es la función modbus (Read Holding Register), 00 7A es la dirección del registro a leer, en este caso 00 7A es 122 en hexadecimal, y por último 00 02 es la cantidad de registros a leer, en este caso 2 por que el registro es de 32 bits.

En la respuesta por parte del servidor (RX), se puede observar que la respuesta es 01 03 04 00 00 00 0F, donde 01 es el ID, 03 es la función modbus (Read Holding Register), 04 es la cantidad de bytes del dato, y 00 00 00 0F es el dato que tiene el registro 122 en este caso es 0F que en decimal es igual a 15.

Para el caso de escritura a registros del servodrive, se debe de seleccionar la función 16 de modbus, encontrada en la barra de herramientas del software *modbus poll*, una vez seleccionada se despliega la siguiente ventana:

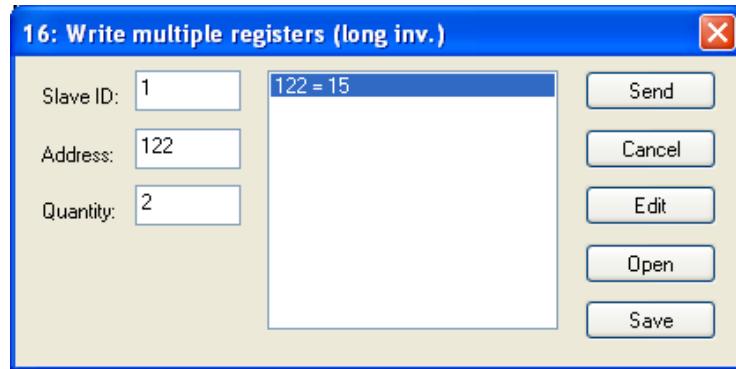


Figura 54: Configuración para escritura de registro

Una vez estando en la ventana, resulta fácil el configurar lo que se requiere realizar, únicamente establecemos la dirección del registro, la longitud de dato del registro y para escribir únicamente le damos en *edit* para poder escribir un nuevo valor.

En la ventana de tráfico de comunicaciones se puede observar la trama que envía por parte del cliente y la respuesta que se recibe por parte del servidor, observando el código de función, que en este caso es 10 hex (16 en decimal):

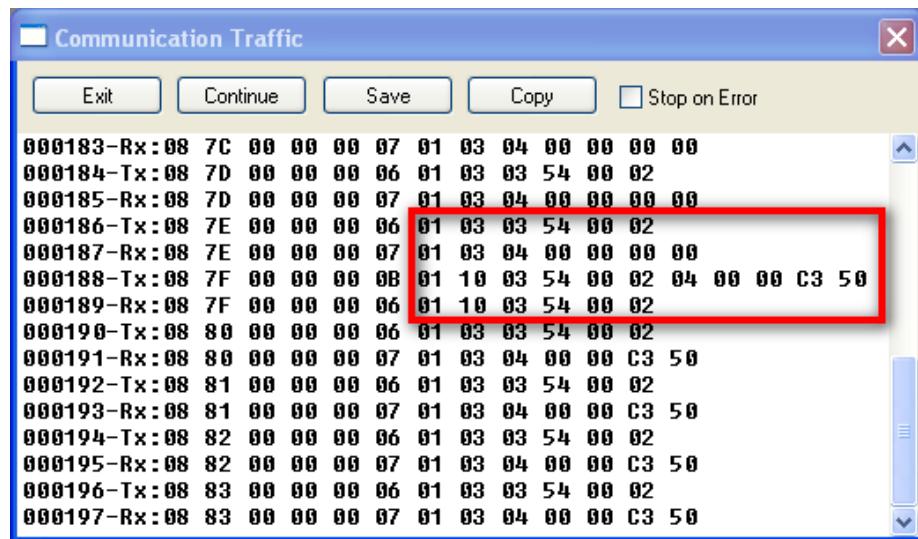


Figura 55: Tráfico de datos en Modbus Poll

Para finalizar con las pruebas de comunicación, se utilizó otra herramienta llamada *WireShark* el cual esta herramienta permite observar los paquetes de datos dentro de una red, observando que los datos enviados entre la computadora y el servodrive realmente es una comunicación puramente modbus TCP/IP:

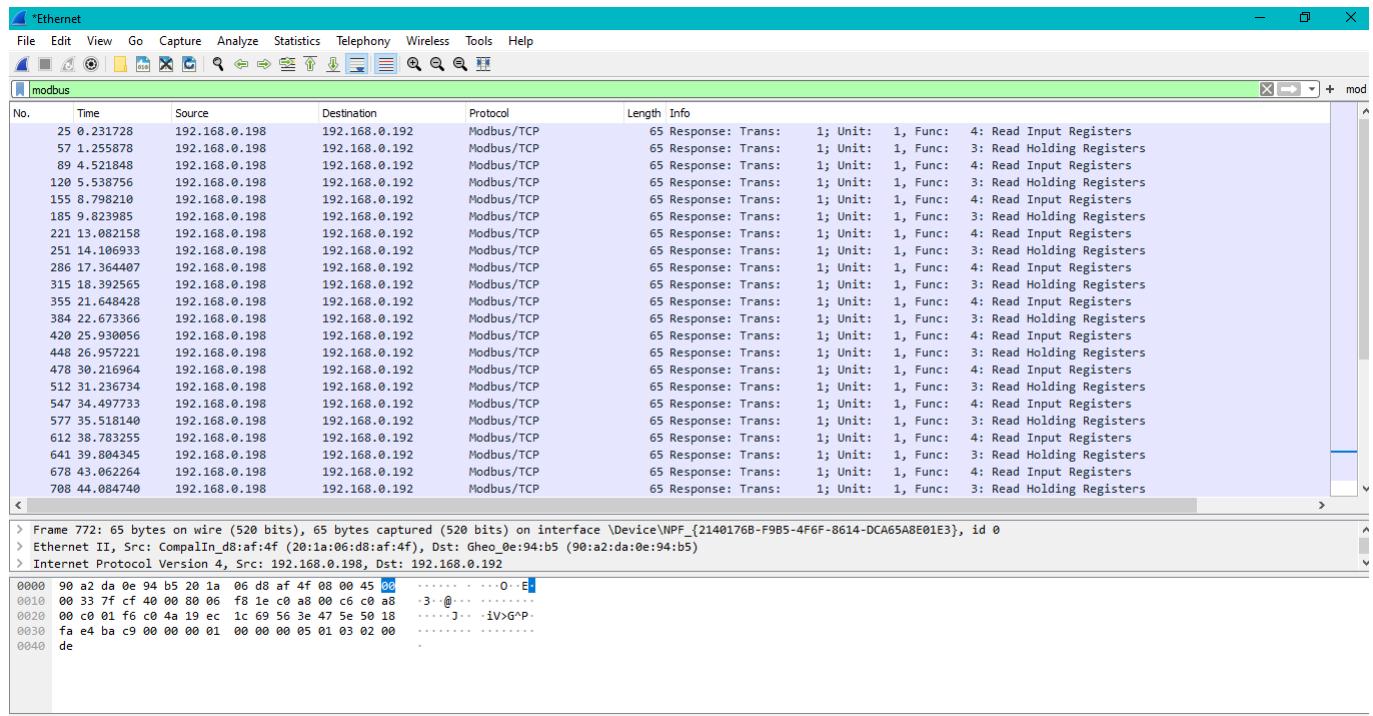


Figura 56: Tráfico de datos mediante WireShark

Con las anteriores pruebas de comunicación realizadas en la red Modbus TCP/IP se comprendió como se envían solicitudes para tomar lectura a registros, así como también como se envían las solicitudes para escribir un registro en específico. Por lo tanto el Arduino debe de ser capaz de leer registros a través de la función 03 “Read holding registers” y escribir registros a través de la función 16 “Write Multiple Registers” que son las funciones que el servodrive acepta para poder hacer cambios en la configuración de sus respectivos espacios de memoria.

#### 10.7.4. Identificación de registros a implementar en proyecto

El paso que sigue es el identificar los registros de configuración y lectura que se deben de contemplar para integrarlos al desarrollo del proyecto. Los registros de memoria que utilizan para la configuración del movimiento del servomotor son los siguientes:

COMANDOS DE AJUSTES DE MOVIMIENTO SERVOMOTOR GUI A CONTROL			
#	Parameter	Dirección	Descripción
1	SM.MODE	751	Tipo de movimiento a ejecutar
2	SM.V1	758	Velocidad 1
3	SM.V2	760	Velocidad 2
4	DRV.ACC	218	Aceleración
5	DRV.DEC	232	Desaceleración
6	SM.T1	755	Temporizador 1
7	SM.T2	757	Temporizador 2
8	DRV.DIR	235	Sentido de giro

Figura 57: Registros de memoria para la configuración del movimiento

Como se puede observar, se utilizan 8 comandos de configuración para poder realizar movimientos del servomotor de acuerdo a lo que el usuario en la interfaz gráfica establezca, por lo tanto la trama diseñada para el proyecto que envía la GUI a Arduino, para que posteriormente se envíen los datos de configuración de parámetros por Modbus TCP/IP a servodrive es la siguiente:



Figura 58: Trama diseñada para el envío de paquete de configuración

Para tomar lectura de valores de registros de memoria del servodrive, únicamente se leen 3 registros, los cuales son:

REGISTROS PARA MONITOREAR EN GUI (ARDUINO TO GUI)			
#	Parameter	Dirección	Descripción
1	VL.FB	858	Velocidad actual
2	DRV.ACTIVE	221	Estado del Driver (activo/desactivado)
3	DRV.HWENABLE	1055	Estado de paro de emergencia

Figura 59: Registros servodrive para monitoreo en GUI

La trama que envía el Arduino a GUI que contiene los datos de los registros de servodrive para monitorear en GUI es la siguiente:

DATOS DE REGISTROS MODBUS A MONITOREAR EN GUI							
VELOCIDAD ACTUAL	Z	STATUS DRIVER	Y	STATUS PARO DE EMERGENCIA	X	DATO TORQUE FUTEK	W

Figura 60: Trama de datos para monitorear en GUI

El último dato de la trama no es un registro modbus, ese dato lo proporciona el sensor encoder Futek TRS605.

Por último, los comandos de control de movimiento y habilitación que se utilizan de GUI a arduino son los siguientes:

REGISTROS DE CONTROL MOVIMIENTO (GUI TO ARDUINO)			
#	Parameter	Dirección	Descripción
1	DRV.EN	254	Habilitación de voltaje de potencia
2	DRV.DIS	236	Deshabilitación de voltaje de potencia
3	SM.MOVE	752	Inicio de movimiento
4	DRV.STOP	274	Detener movimiento

Figura 61: Registros de control de movimiento y habilitación

Estos últimos registros no necesitan enviarse a través de una trama de datos, ya que estos se utilizan cada vez que el usuario hace click en el botón específico.

Como se puede observar, en total suman 15 registros del servodrive que se acceden a través de modbus TCP/IP teniendo una plataforma integrada entre la GUI y el sistema de control, logrando así controlar el servomotor de acuerdo a la configuración realizada en la GUI.

#### 10.7.5. Arduino Modbus TCP/IP

Una vez que se conocen los registros que serán necesarios implementar en proyecto y que también se sabe que tipo de funciones modbus son permitidas en los registros del servodrive, se debe de revisar la documentación de la librería Modbus TCP/IP para conocer el sintaxis correcto para mandar la trama Modbus TCP/IP hacia el servodrive.

En el propio código del desarrollo de la librería muestra:

```
MB_FC_NONE          = 0,
MB_FC_READ_COILS   = 1,
MB_FC_READ_DISCRETE_INPUT = 2,
MB_FC_READ_REGISTERS = 3,
MB_FC_READ_INPUT_REGISTER = 4,
MB_FC_WRITE_COIL    = 5,
MB_FC_WRITE_REGISTER = 6,
MB_FC_WRITE_MULTIPLE_COILS = 15,
MB_FC_WRITE_MULTIPLE_REGISTERS = 16
```

Figura 62: Funciones Modbus que soporta librería

Por lo tanto, las funciones modbus que se deben de utilizar para leer registros del servodrive Kollmorgen son:

- MB\_FC\_READ\_REGISTER (Función 03 Modbus)
- MB\_FC\_WRITE\_MULTIPLE\_REGISTERS (Función 16 Modbus)

La sintaxis que se realiza en código de arduino para leer un registro modbus es la siguiente:

```
Mb.MbData[0] = 0;
Mb.Req(MB_FC_READ_REGISTERS, 1055,1,0); //DRV.HWNABLE
```

Figura 63: Leer registro 1055 (DRV.HWNABLE)

Donde después de invocar la función MB\_FC\_READ\_REGISTERS, se debe de indicar cual es la dirección del registro, en este caso 1055, la cantidad de bytes que va a leer, en este caso 1 byte, y el tercer parámetro de función se deja en 0.

Para la escritura de un registro del servodrive, se debe de realizar la siguiente sintaxis:

```
Mb.MbData[0] = num2;
Mb.Req(MB_FC_WRITE_MULTIPLE_REGISTERS, 758,1,0); //SM.V1
```

Figura 64: Escribir en registro 758 (SM.V1)

Por lo tanto, con base a lo anterior ya se puede realizar código de control en el Arduino para la lectura y la escritura de los registros que se van a utilizar en el proyecto. La red local que se crea para el proyecto es la siguiente, dejando la computadora parte de la red, ya que la computadora se puede necesitar para depuración de errores, aunque se puede retirar de la red.

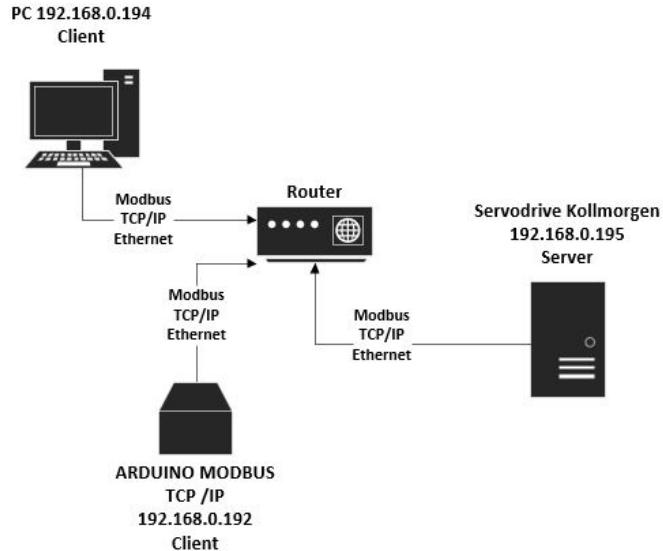


Figura 65: Diagrama de red implementada para el proyecto

Una parte del código de control para la lectura de los registros de monitoreo para mandar a la GUI es la siguiente:

```

316 void Poller(){ /*secuencia de modbus registros de monitoreo: velocidad,Driver En, emergency Stop
317     delay(1);
318     acc = acc + 1;
319
320     if(acc ==120){ //!Mete ruido cuando se coloca un tiempo pequeño en acc
321         Mb.MbData[0] = 0;
322         Mb.MbData[1] = 0;
323         Mb.Req(MB_FC_READ_REGISTERS, 858,2,0); /* Solicitud de lectura registro VL.FB 858
324     }
325     if(acc == 240){
326         MSB_W_VL = Mb.MbData[0];
327         LSB_W_VL = Mb.MbData[1]; /*Tomar dato de Velocidad motor
328     }
329
330     if(acc == 360){
331         Mb.MbData[0] = 0;
332         Mb.Req(MB_FC_READ_REGISTERS, 221,1,0); /*Solicitud de lectura registro DRV.ACTIVE 221
333     }
334     if(acc == 480){
335         STATUS_drv = Mb.MbData[0]; /*Tomar dato de Status driver
336     }
337

```

Figura 66: Parte del código de control para lectura de registros Servodrive

Parte del código de control para la escritura de registros de configuración es la siguiente:

```
532 void Poller_3(){ /*Configurar registros de servodrive modbus
533     delay(1);
534     acc = acc + 1;
535
536     if(acc ==120 ){
537         Mb.MbData[0] = num1;
538         Mb.Req(MB_FC_WRITE_MULTIPLE_REGISTERS, 751,1,0); /*Escribir en registro SM_MODE 751 (tipo de movimiento)
539     }
540
541     if (acc == 240){
542         Mb.MbData[0] = num2;
543         Mb.Req(MB_FC_WRITE_MULTIPLE_REGISTERS, 758,1,0); /*Escribir en registro SM.V1 758 (Velocidad 1)
544     }
545
546     if (acc == 360){
547         Mb.MbData[0] = num3;
548         Mb.Req(MB_FC_WRITE_MULTIPLE_REGISTERS, 760,1,0); /*Escribir en registro SM.V2 760 (Velocidad 2)
549     }
550
551     if (acc == 480){
552         Mb.MbData[0] = num4;
553         Mb.Req(MB_FC_WRITE_MULTIPLE_REGISTERS, 218,1,0); /*Escribir en registro ACC 218 (límite de aceleración, no viene en manual)
554 }
```

Figura 67: Parte del código de la escritura de registros servodrive

Una limitante que se observa en el código de control para la comunicación Modbus TCP/IP es que para cada sentencia de registro, sea para escritura o lectura, esta sincronizada con un tiempo, el cual si se necesita trabajar con varios registros de modbus en secuencia, este tiempo puede superar más de un segundo en completarse, por lo cual es un tiempo que debe de tomarse en cuenta, ya que este tiempo se traduce como un retraso para la aplicación.

Se deben de contemplar únicamente registros que si sean necesarios para el proyecto y optimizar los tiempos entre solicitudes de cada registro.

Una manera de optimizar los tiempos, es observar si existen direcciones de registro subsecuentes (adyacentes), y poder eliminar algunas líneas de código, ahorrando tiempos de ejecución de instrucción.

Esta optimización de líneas de ejecución se implementó para leer el registro 858 y 859 que es el dato de velocidad VL.FB, el cual al ser dos direcciones subsecuentes, se eliminaron algunas líneas para ahorrar tiempo, y el bloque de código quedó de la siguiente manera:

```
Mb.MbData[0] = 0;
Mb.MbData[1] = 0;
Mb.Req(MB_FC_READ_registers, 858,2,0); // * Solicitud de lectura registro VL.FB 858
}
if(acc == 240){
    MSB_W_VL = Mb.MbData[0];
    LSB_W_VL = Mb.MbData[1]; /*Tomar dato de Velocidad motor
```

Figura 68: Bloque de código optimizado para lectura de registro modbus

Como se observa, se declaran dos métodos MbData, estos para tomar los dos valores de cada registro. En la parte de la sintaxis de la función READ REGISTERS, se observa que en el segundo parámetro tiene un valor de 2, esto es porque se está configurando la función que lea dos espacios de memoria (2 registros), iniciando desde la dirección 858, por lo cual en el método MbData[0] guarda el valor del registro 858 y en MbData[1] guarda el valor del registro 859.

Al final de cuentas, son dos registros de 16 bits, por lo cual forman una palabra de 32 bits. Si se busca en el manual de usuario el parámetro VL.FB se encontrará que este dato está representado por 4 registros de 32 bits, por lo tanto es un registro completo de 64 bits, pero para el proyecto únicamente nos interesa la parte baja de esta palabra de 64 bits, que son los dos registros últimos (858, 859) la parte LSB de la palabra, el cual entre estos dos almacenan valores de velocidades bajas.

Se toman únicamente los dos registros LSB de la palabra ya que el banco de pruebas no va a superar velocidades por arriba de 1.500 RPM, por lo tanto con los dos registros LSB es más que suficiente para obtener el dato que necesita la aplicación.

## 10.8. Sensor de torque Futek TRS605

En este apartado se desglosa el desarrollo para la implementación del sensor Futek TRS605.

### 10.8.1. Dato Velocidad

La forma en que se presenta el dato de velocidad proveniente del sensor Futek TRS605, es mediante señales cuadradas, el cual es mediante el desfase y el ancho de pulso de estas señales las que determinan a que velocidad esta girando el eje y en que sentido de dirección se encuentra.

La manera en que se leen este tipo de señales es mediante interrupciones del microcontrolador, funcionando como si fuera un frecuencímetro. Una vez que se obtiene el valor de diferencia entre señales, el resultado pasa a ser procesado para convertirlo en la representación de velocidad.

Los pines usados para interrupciones en el arduino Mega (Atmega2560) fueron los pines A13 y A14.

Además de que se utilizó la librería *RotaryEncoder*.

Una vez instalada la librería se realizan pruebas de funcionamiento para obtener el dato de velocidad en el monitor serie, el cual se logra satisfactoriamente.

Parte del código del ejemplo es el siguiente:

```
// Read the current position of the encoder and print out when changed.
void loop()
{
    static int pos = 0;
    static int milis=0;//guarda el instante
    float deltaT,deltaPos,angSpeed;
    int newmilis=0;//inicializa nuevo instante en cero;
    int newPos = encoder.getPosition(); //Toma valor de posición del encoder

    if (pos != newPos) {//si hubo un cambio en la cuenta, entonces:
        newmilis=millis();
        deltaPos=newPos-pos;//calcular el cambio en cuentas,

        if(deltaPos){//procesar solo deltas positivos, skipeando los overflows
            deltaT=((newmilis-milis)/1000); //deltat en segundos
            deltaPos=deltaPos/360; //deltapos en revoluciones ?
            angSpeed=deltaPos/deltaT; //
            Serial.print(angSpeed,3);
            Serial.println();
            delay(1000); //lanzar lecturas solo cada segundo
        }
        milis=newmilis;
        pos = newPos;
    }
}
```

Figura 69: Parte del código del encoder Futek para obtener dato de velocidad

La dificultad que se encontró al realizar la implementación del sensor Futek para obtener el dato de velocidad, fue que la librería propiamente esta desarrollada para la tarjeta Arduino UNO (Atmega328p), no para Arduino Mega (Atmega2560).

Por lo tanto al principio no se obtenían resultados en el programa, se realizó un estudio e investigación para poder modificar propiamente la librería y que sea posible utilizar esta librería de encoder en el Arduino Mega.

Se llega a una parte del desarrollo de la librería donde establece los registros directos del microcontrolador, donde ahí es donde configura de manera manual los registros de interrupción.

Se investiga y se encuentran los registros de interrupción del microcontrolador Atmega2560 y se configuran adecuadamente en ese bloque de código, como se observa:

```
// Registros directos del microcontrolador para habilitar interrupciones
PCICR |= (1 << PCIE2);
PCMSK2 |= (1 << PCINT21) | (1 << PCINT22);
```

Figura 70: Registros del microcontrolador AtMega2560 para interrupciones

Haciendo lo anterior, se logran habilitar las interrupciones y la librería funciona ahora en el Atmega2560.

### 10.8.2. Dato torque

Para obtener el dato del torque enviado propiamente por el sensor Futek, fue necesario dar uso al módulo ADC ADS1115, ya que la señal que representa el dato del torque esta representada bajo señal analógica diferencial, que como se vio anteriormente en la sección de reconocimiento de dispositivos, esta señal analógica diferencial no puede ser leída por arduino directamente ya que contiene voltajes negativos y positivos. Es por ello que se toma como solución este módulo ADC el cual cuenta con modo diferencial.

El mapa de conexiones que se realizaron fueron las siguientes:

<b>ADS1115</b>	<b>Conecta con:</b>
A0	Vout + encoder (Green)
A1	Vout - encoder (White)
VDD	5V+ Arduino
GND	Arduino GND
SCL	SCL 21 Arduino
SDA	SDA 20 Arduino

Figura 71: Conexionado del módulo ADS115

Una vez teniendo el módulo conectado directamente, se hace uso de la librería *Adafruit ADS1X15* y *wire*, la primera librería es la librería propia del módulo y la segunda librería es para habilitar comunicación I2C.

Dentro del código, se necesita que inicializar el módulo ADS1115 de la siguiente manera:

```
/*Factor de escala ADS1115
ads.setGain(GAIN_SIXTEEN);

/*Iniciar el ADS1115
ads.begin();
```

Figura 72: Configuración e inicialización del módulo ADS1115

Una vez realizado lo anterior, solamente se debe de ejecutar el siguiente comando para obtener el valor de la señal analógica diferencial, una vez que se obtenga el dato, se debe de convertir el dato a la representación de mV, esto ya se hace en el código de desarrollo.

```
ads.readADC_Differential_0_1();
```

Figura 73: Obtener lectura de la señal analógica

Para mejorar la estabilidad en las lecturas de señal, implementé un filtro para poder promediar las lecturas y estabilizar aún más las lecturas:

```
for(int i = 0; i < MeasurementsToAverage; ++i) /*filtro software promediar lecturas
{
    PromLectura += ads.readADC_Differential_0_1();
    delay(1);
}
PromLectura /= MeasurementsToAverage;
PromLectura *= factorEscala;
```

Figura 74: Filtro mediante software para estabilizar lecturas de señal

Se concluye con la integración del sensor Futek TRS605 al proyecto de residencia.

## 10.9. Freno Industrial Warner Electric

Para integrar este dispositivo al proyecto se debe de contemplar dos salidas digitales para poder hacer la siguiente tabla lógica:

### Mode Options

#### Single Mode

Ch1	Ch2	Ch2 Ove	Ch1 output
0	0	0	No change
0	0	1	On at full torque
0	1	0	Off
0	1	1	On at full torque
1	0	0	On at adjusted torque
1	0	1	Full torque
1	1	0	On at adjusted torque
1	1	1	Full torque

Figura 75: Tabla lógica de control Freno Warner Electric

En este caso, solamente serán utilizadas dos salidas digitales del arduino Mega, la salida CH1 y Salida CH2. Pero al estar realizando pruebas se encuentra que el nivel lógico que necesita el controlador del freno Warner Electric debe de ser entre 8V y 15 V por lo cual el arduino no será capaz de mandar esos niveles lógicos directamente de sus salidas.

Es por ello que se diseña un circuito básico para la conversión del nivel de voltaje lógico, de 5V a 12V aproximadamente:

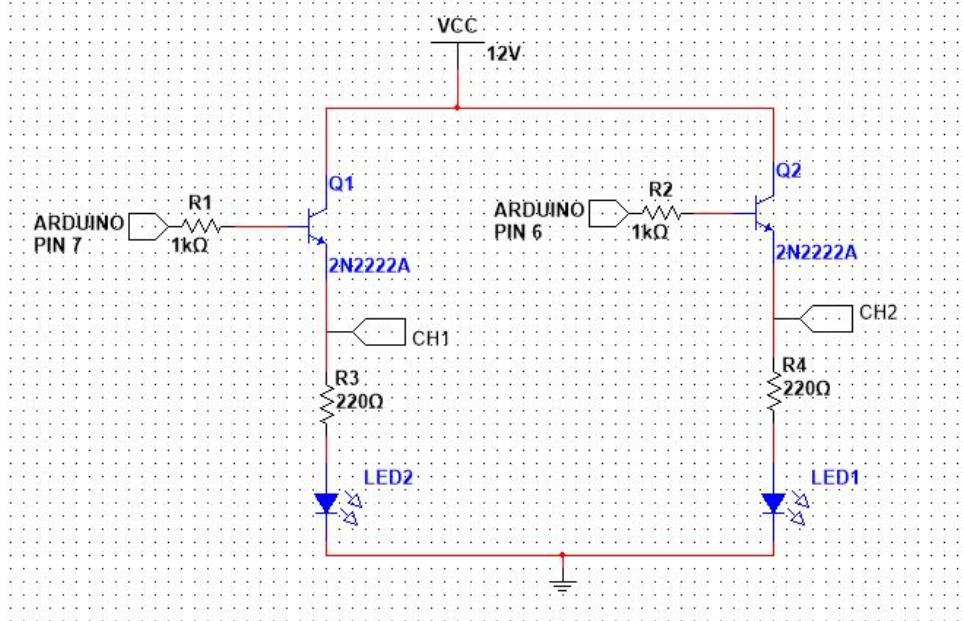


Figura 76: Circuito de conversión nivel lógico 5V - 12V

Con esta solución se obtienen las salidas de accionamiento del propio Arduino, trabajando con 5V y habilitando el transistor correspondiente, y permitiendo una caída de tensión sobre Colector Emisor y tomando un voltaje de aproximado 11 V, siendo este voltaje capaz de reconocer como '1' lógico y accionar el freno de acuerdo a la tabla lógica.

El código implementado para este dispositivo resulta fácil, solamente es definir las salidas del arduino y simplemente mandarlas a habilitar cuando la aplicación requiera hacer un cambio en el estado del freno industrial Warner Electric.

```
/*salidas digitales para freno industrial
#define CH1 7
#define CH2 6
```

Figura 77: Definición de salidas para Freno Warner Electric

```
else if (inputString == "M2"){ /*Activar freno ajustado
digitalWrite(CH1, HIGH);
tiempo(300);
digitalWrite(CH1, LOW);
tiempo(150);
inputString = "";
}
else if (inputString == "N2"){ /*Activar freno libre
digitalWrite(CH2, HIGH);
tiempo(300);
digitalWrite(CH2, LOW);
tiempo(150);
inputString = "";
}
```

Figura 78: Código implementado para el Freno Warner Electric

Como se observa, la lógica de activación para cambiar los modos del freno industrial es de manera sencilla y se sigue la tabla lógica que se presentó al inicio, logrando así el cambio de modos de operación del freno de acuerdo a lo que necesite la aplicación a través de la GUI de usuario.

## 10.10. Desarrollo de GUI

Para comenzar con el desarrollo de la GUI, se establece como IDE o plataforma de desarrollo *Visual Studio Community* el cual es una plataforma libre y gratuita para desarrollo de proyectos sin fines de lucro, por lo cual se comienza a desarrollar el código de la GUI a través de esta plataforma.

La manera de desarrollar la GUI en esta plataforma no es para nada difícil ya que solamente debes de tener conocimientos básicos sobre Windows Form (el cual esta característica la proporciona el framework Windows .net ), se debe de tener en mente que Windows Form son ventanas de aplicación el cual cada ventana tiene sus eventos, métodos y atributos las cuales se deben de configurar/programar para que la ventana trabaje de acuerdo a lo que debe de realizar.

Cuando se tiene varias ventanas y estas se van abriendo de acuerdo de como vaya ejecutándose el programa de aplicación se verá una aplicación dinámica y fluida, como si fuera una aplicación de escritorio normal que se abre en la computadora.

Se debe de tomar en cuenta que la aplicación de desarrollo que se pretende implementar debe de tener comunicación serial para que se pueda comunicar al puerto COM del arduino , ya que esa será la interface entre GUI y el sistema de control, tal cual somo se vió en el diagrama de bloques que se presentó al inicio.

Hablar del código del a GUI es muy extenso, ya que se tienen 4 bloques de código que trabajan en conjunto para poder crear la aplicación completa, además que tanto el desarrollo del sistema de control como el desarrollo de la interfaz gráfica, CIATEQ no permite que se haga público el proyecto entero, esto por que se prende cambiar de plataforma y hacer el proyecto privado.

Por último, se agrega la primer interfaz gráfica que se desarrolló para comenzar a realizar pruebas de funcionamiento, la versión final se verá en la parte de *resultados*.

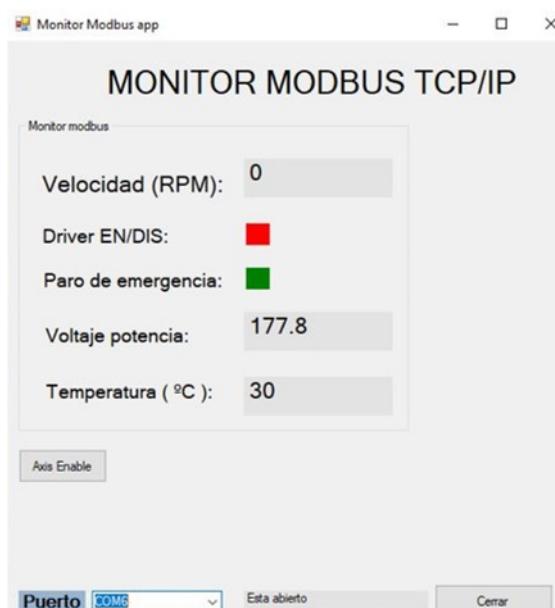


Figura 79: Primer GUI beta del proyecto de residencia

## 11. Resultados

En esta sección se abordan los resultados obtenidos con el desarrollo del proyecto de residencia.

### 11.1. Inicialización de aplicación y pantalla principal

La aplicación se abre a través del ícono que se encuentra en el escritorio:

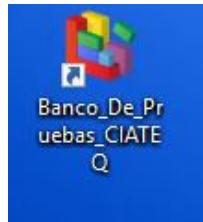


Figura 80: Ícono de la aplicación

La imagen siguiente muestra la pantalla principal de la aplicación:



Figura 81: Pantalla principal de la aplicación

En esta ventana se debe de seleccionar el puerto COM del arduino para enlazar la GUI con el sistema de control. Una vez seleccionado el puerto correspondiente del arduino, dar click en *Open*

También se observa que el usuario puede seleccionar la velocidad de baudios, en este caso la aplicación de control se encuentra con una velocidad de 9600, por lo cual se deja por default.

Si la comunicación fue exitosa, se desplegará un mensaje en color verde, caso contrario será de color rojo:

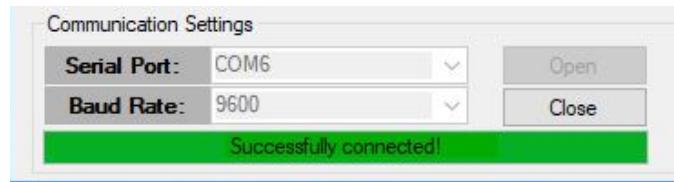


Figura 82: GUI se conectó satisfactoriamente con sistema de control



Figura 83: GUI no logró conectarse con el sistema de control

Cuando se logra una comunicación exitosa, se habilitará los modos de operación del banco de pruebas, como se muestra:



Figura 84: Habilitación de modos de operación banco de pruebas

Para ingresar al modo deseado, únicamente se debe de seleccionar el modo de operación a donde desea ingresar, y presionar el botón *Ok*



Figura 85: Selección de modo de prueba

Como se observa, la GUI es inteligente, cuando el usuario selecciona el modo de operación, el otro inmediatamente se bloquea, si se deselecciona la casilla, vuelve a estar habilitado.

En esta ventana también se puede dar uso al botón *Close* el cual cierra el puerto serie.

La ventana puede cerrarse dando uso al botón *Cancel* o dando uso al botón de *cerrar ventana*.

#### 11.1.1. Carpeta de recopilación de datos

Cuando la aplicación de escritorio se ejecuta, crea una carpeta general llamada *Datos pruebas* en el disco local C, esta carpeta la crea si no existe, y no la crea si ya existe en la ruta. En esta carpeta se almacenarán los datos recopilados de cada modo de operación.

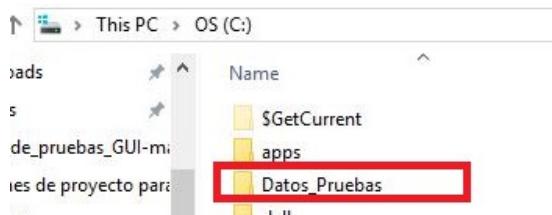


Figura 86: Carpeta general “Datos pruebas” en disco local C

Al ingresar dentro de esta carpeta, se observa que existe una carpeta de sesión, el cual cuenta con la fecha hora del inicio de sesión (que es cuando se ejecutó la aplicación):

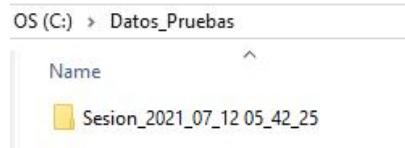


Figura 87: Carpeta de sesión

Dentro de esta carpeta se van creando dos subcarpetas, de acuerdo a lo que se realizó dentro de esa sección. Si el usuario ejecutó pruebas en los dos modos de operación se crean las siguientes carpetas:

OS (C:) > Datos_Pruebas > Sesion_2021_07_12_05_42_25	
Name	Date modified
Modo Generador	12/07/2021 05:54 ...
Modo Motor	12/07/2021 06:03 ...

Figura 88: Carpeta de sesión

Si solamente el usuario ingresó a un solo modo de operación, entonces solamente se va a crear una carpeta, incluso si el usuario ejecuta la aplicación, pero no entró a ningún modo de operación y cierra la aplicación, no se creará ninguna carpeta en esta ruta, puesto que no se ejecutó ninguna prueba en ningún modo de operación.

Dentro de estas carpetas se encontrarán los datos que el usuario guarde cuando se encuentre ejecutando pruebas en algún modo de operación, más adelante se observará el contenido de estas carpetas.

## 11.2. Modo Motor

Cuando se ingresa al modo motor, se despliega la siguiente ventana:

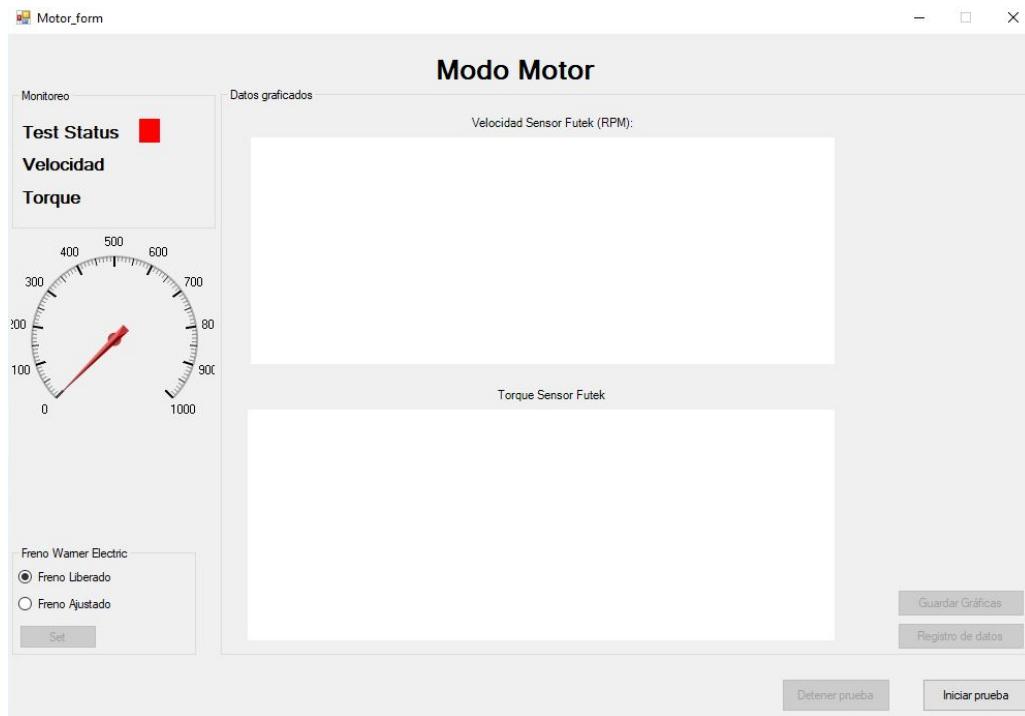


Figura 89: Ventana del modo motor

Se puede apreciar en esta ventana un control dedicado para el modo del freno Warner Electric, puede ser configurado en modo *Freno liberado* o *Freno ajustado*, únicamente se selecciona el modo deseado y se oprime el botón *Set*.

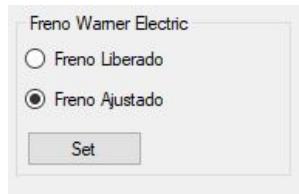


Figura 90: Cambiar de modo el Freno Industrial Warner Electric

Como se vio en la definición del proyecto, este modo de operación es una pantalla de monitoreo para observar las señales de respuesta que nos entrega el sensor Futek TRS605, además de eso, en modo offline se podrá cambiar el modo de operación del freno, sea en modo ajustado o en modo liberado.

Para comenzar con la prueba, únicamente se debe de oprimir en el botón *Iniciar prueba* para comenzar a graficar las señales del sensor Futek:

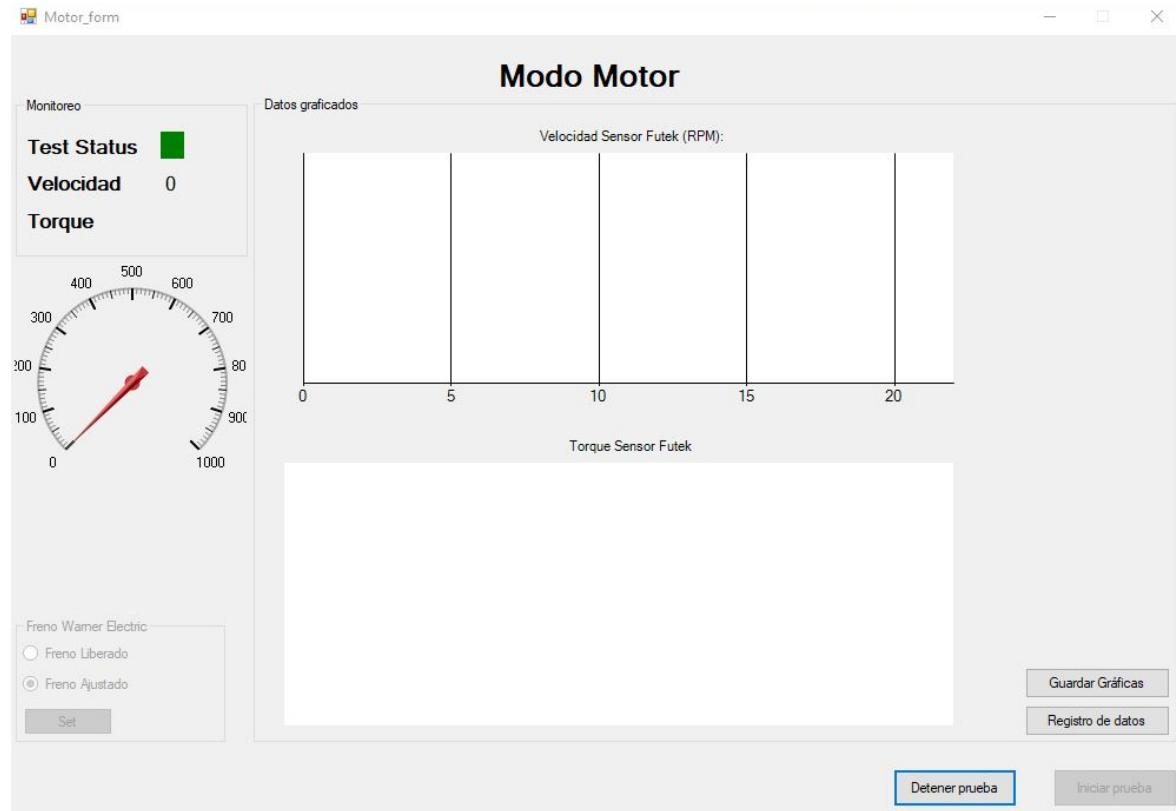


Figura 91: Inicio de prueba modo motor

Se observa que el control del freno Warner Electric ha sido bloqueado, y que dos botones *Guardar gráficas* y *Registro de datos* han sido habilitados.

Conforme el motor que esta bajo prueba comience a generar movimiento, se irán gráficando los datos de respuesta del sensor:

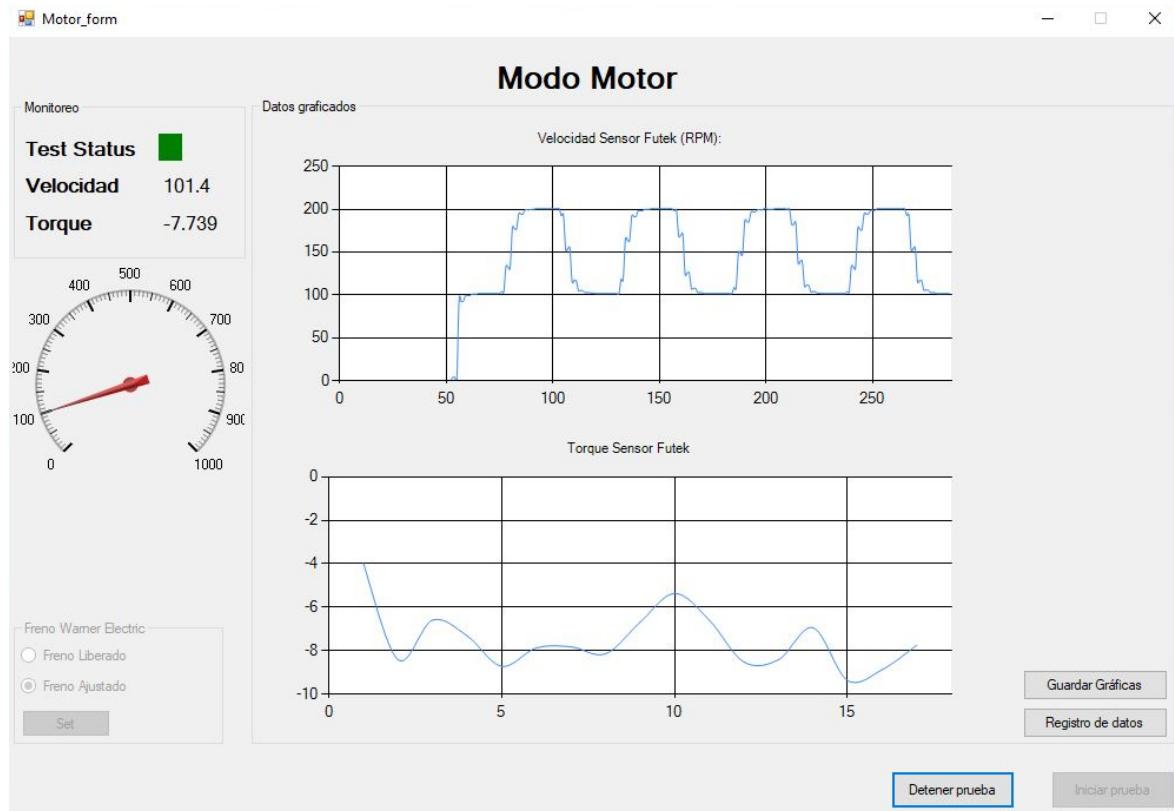


Figura 92: Graficando valores de sensor Futek TRS605

Cuando se oprime el botón *Iniciar prueba*, esta se mantiene activa , graficando cualquier valor que el sensor Futek proporcione, y si el motor bajo prueba ya no se encuentra realizando ningún movimiento, la prueba continuará activa hasta que se presione el botón *Detener prueba*, una vez presionando ese botón se deja de graficar los datos.

Una forma de comprobar si la prueba esta activa o no, es por el indicador de color que está en el grupo llamado *Monitoreo*, ese indicador tendrá color rojo cuando la prueba esta detenida y color verde cuando la prueba se encuentra iniciada.

También en ese mismo grupo de monitoreo, se muestra numéricamente los valores de velocidad y torque.

Cuando se presiona el botón *Guardar Gráficas* aparece el siguiente aviso:

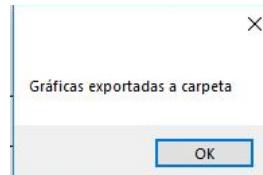


Figura 93: Aviso de exportación de gráficas

Si se presiona el botón *Registro de datos* se despliega la siguiente ventana:

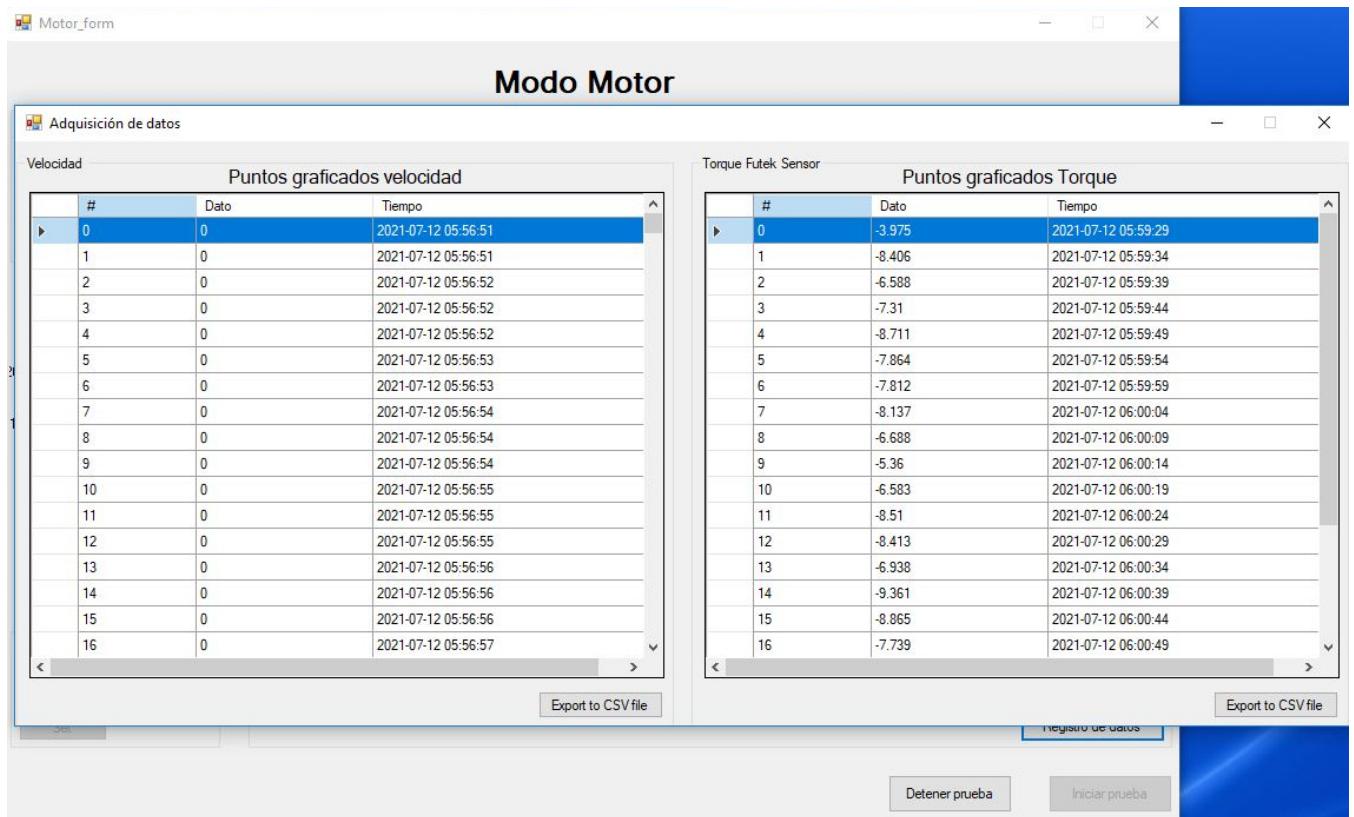


Figura 94: Ventana de registro de datos

Se puede observar los registros de datos almacenados en una pequeña base de datos, tanto los datos de velocidad como los datos de torque.

Como estos datos también son de interés para el usuario, se pueden exportar a excel en formato CSV, únicamente se debe de presionar el botón respectivo de cada base de datos y aparecerá el siguiente aviso:

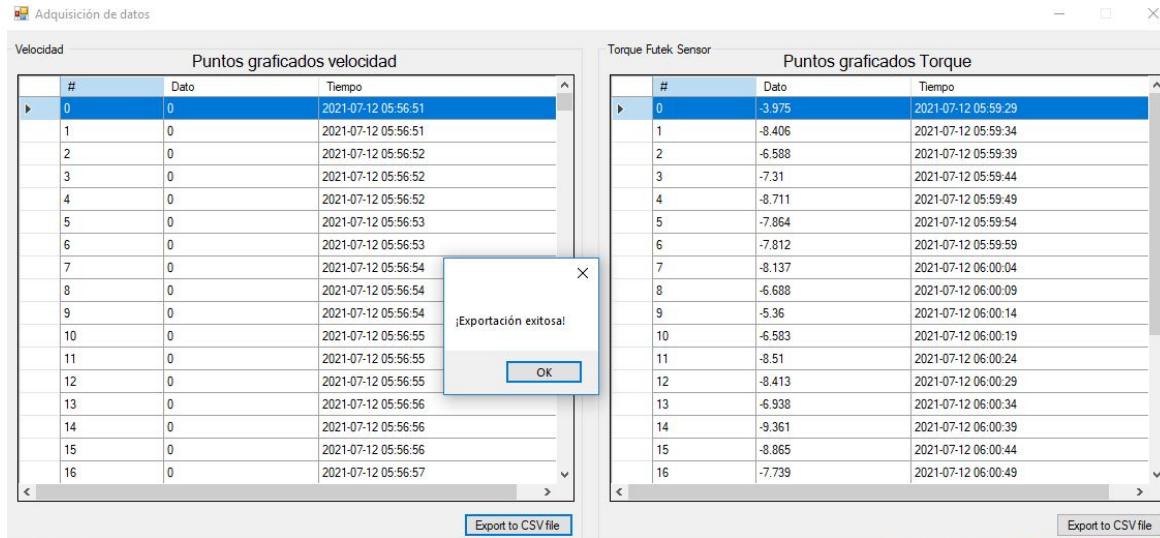


Figura 95: Exportación de datos

Los datos de prueba que el usuario exporte, serán almacenados dentro de la carpeta *Modo Motor* en la ruta de la carpeta *Datos pruebas* como se explicó en el apartado *Carpeta de recopilación de datos*.

Por lo tanto, al acceder a la carpeta *Modo Motor*, se encontrarán los siguiente datos de prueba almacenados:

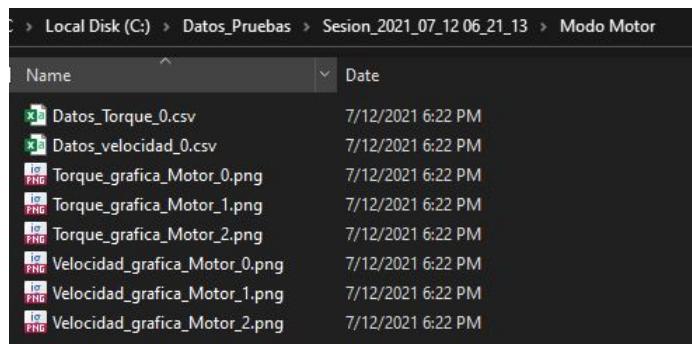


Figura 96: Datos de prueba almacenados en carpeta motor

Como se puede observar, los datos del registro de la base de datos de la aplicación son almacenados directamente en un archivo .csv listo para ser observados en excel y empezar con el análisis de datos. Las gráficas por su parte son imágenes en formato .png

Cabe señalar que tanto los archivos .csv como los archivos de imagen .png no serán sobreescritos si se vuelve a exportar los datos, ya que al final del nombre del archivo se le agrega un valor numérico para poder crear otros archivos con diferente id.

Por seguridad, al querer salir de la ventana principal del modo motor, aparecerá el siguiente mensaje:

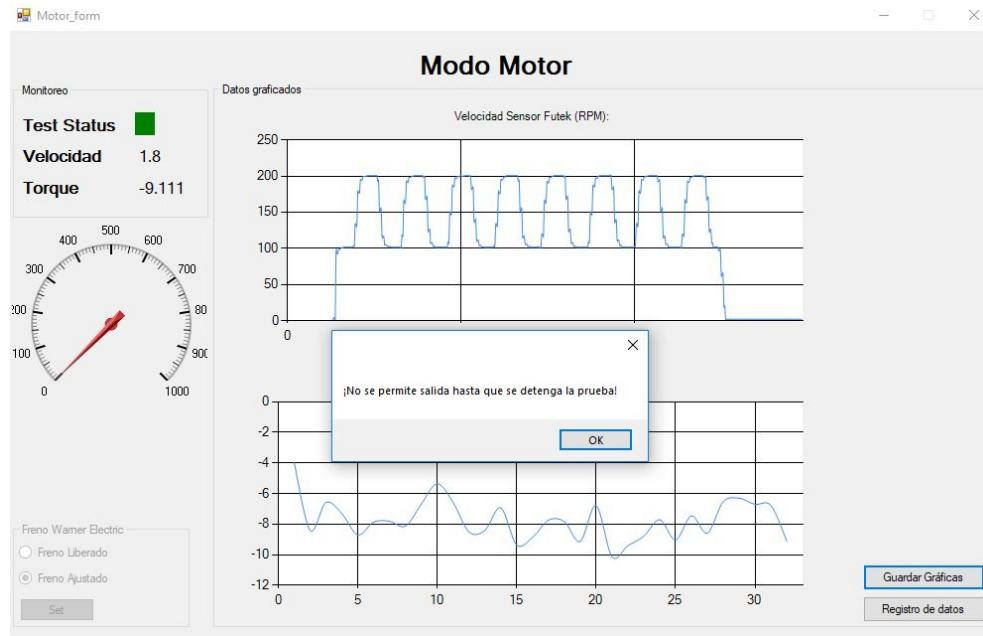


Figura 97: Salida no permitida cuando la prueba esta activa

Para poder salir del modo motor, la prueba debe de estar detenida, esto se logra al presionar el botón *Detener prueba*, una vez que se presione el botón la prueba se detiene, manteniendo los datos gráficos y los registros de datos.

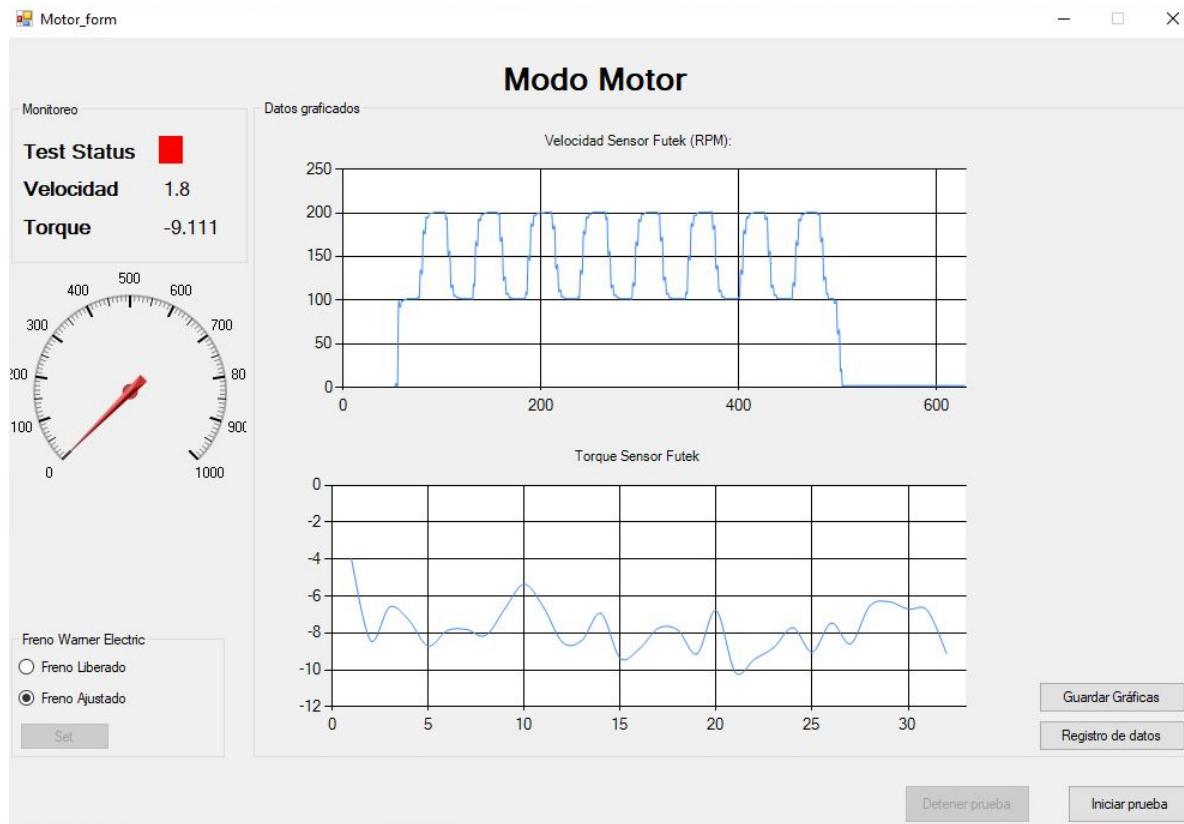


Figura 98: Prueba terminada

Una vez que se encuentra detenida la prueba, el usuario puede cambiar el modo del freno Warner Electric y volver a reiniciar la prueba o también puede salir del modo motor para regresar a la ventana principal.

### 11.3. Modo Generador

Una vez que regresamos a la ventana principal, ahora vamos a ingresar al modo generador de igual forma como lo hicimos con el modo motor.

Cuando se ingresa al modo generador, la ventana que aparecerá en primer instancia será la pestaña *Service motion*:

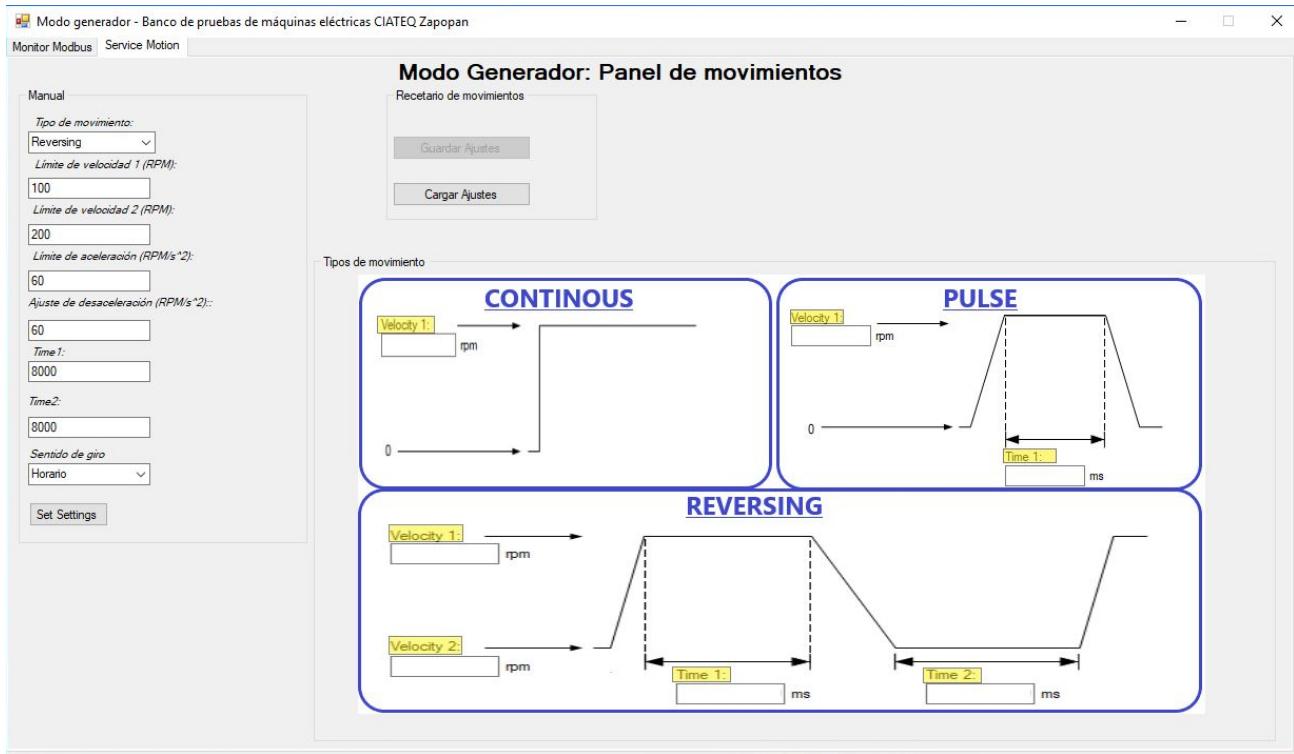


Figura 99: Ventana principal modo generador

En esta pestaña el usuario debe de configurar todos los parámetros de configuración del servomotor, ya que resulta primordial configurar de inicio el servomotor antes de poder iniciar movimientos.

Tomando en cuenta lo anterior, si el usuario intenta ir a la pestaña denominada *Monitor Modbus* donde es ahí donde es posible iniciar movimientos del servomotor, le saldrá un aviso y se bloquea el acceso a esa pestaña:

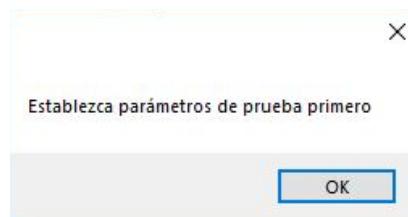


Figura 100: Bloquear acceso a Monitor Modbus

### 11.3.1. Configuración en pestaña “Service motion”

Como se puede apreciar en la pestaña *Service motion* se tiene una ilustración detallada de los tipos de movimiento que el usuario puede seleccionar, en ella se muestra como es la forma del tipo de movimiento y los ajustes que puede configurar el usuario para cada tipo de movimiento.

En el grupo *Manual* es donde el usuario debe de seleccionar y añadir valores numéricos para poder configurar adecuadamente el servomotor.

En primer instancia se tiene una lista desplegable del *Tipo de movimientos* en esta lista se encuentran los tipos:

- **Continuous**
- **Pulse**
- **Reversing**

Que de acuerdo al tipo de movimiento que el usuario seleccione, se habilitarán o deshabilitarán campos de configuración para dicho tipo de movimiento que se ha seleccionado.

En los límites de *velocidad 1 y 2*, el usuario debe de ingresar un dato numérico, el cual este dato tiene un rango, entre 50 a 1500, el cual es la velocidad en RPM que va ejercer el servomotor. Por precaución, si el rango es superado, se pone un valor numérico por default (la velocidad más baja) para prevenir accidentes.

En los campos de *aceleración y desaceleración* sucede lo mismo que con las velocidades, aquí el usuario debe de ingresar los valores numéricos que desea para dichos parámetros, con la condición de que no pueden superar las velocidades que ha asignado anteriormente el usuario esto para prevenir datos ilógicos, por ejemplo si el usuario ingresa una velocidad de 150 RPM y una aceleración de 850 , suena algo ilógico y puede suceder algún accidente, por lo cual se estableció esta condición para que los datos sean lógicos.

En los campos de *timer1 y timer2* son los tiempos que los tipos de movimiento utilizan para ejecutar la prueba, estos de igual forma el usuario debe de ingresar el dato numérico el cual rango es de 100 mS a 60000 mS (60 segundos).

Por último se tiene otra lista desplegable, la denominada *Sentido de giro*, esta lista solo contiene dos elementos:

- **Horario**
- **Antihorario**

Como es de suponer, aquí el usuario debe de establecer el sentido de giro que desea para el servomotor.

Por otro lado, en el grupo *Recetario de movimientos* se observan dos botones:

- *Guardar ajustes*
- *Cargar ajustes*

Se observa que en primer instancia se tiene el botón *Guardar ajustes* deshabilitado y el botón *Cargar Ajustes* se encuentra habilitado.

El botón *Cargar ajustes* le permite al usuario cargar un archivo de configuración previa que realizó en una sesión anterior a la que en este momento nos encontramos, ese archivo de configuración lo crea el botón *Guardar Ajustes*, el archivo que se crea es un text file y contiene todos los datos y selecciones que el usuario realizó en el grupo *Manual*, el cual con ese archivo y con el botón *Cargar Ajustes*, vuelve a dejar todo el grupo *Manual* como lo dejó aquella vez el usuario.

Únicamente el usuario debe de confirmar configuración presionando el botón *Set Settings*, este botón envía la cadena de datos de configuración (frame) al sistema de control (Arduino), y el sistema de control envía la configuración al servodrive mediante Modbus TCP/IP.

El botón *Set settings* también ejecuta una validación de datos de todos los campos, el cual revisa que los datos ingresados sean válidos para la configuración, de tener algún dato incorrecto, saldrá un mensaje en pantalla diciendo *Error en la entrada de datos* y el usuario debe de revisar cual es el dato erróneo.

Volviendo al grupo *Manual*, una vez que el usuario configuró todo lo que necesita para la prueba, el usuario debe de presionar el botón *Set settings* para que comience la validación de datos y si todo es correcto mandé la configuración al servodrive.

Cuando son válidos los datos que ingresó el usuario, el botón *Guardar ajustes* se habilita y el usuario puede guardar si lo desea los parámetros que definió anteriormente. Cuando se presiona saldrá el siguiente aviso:

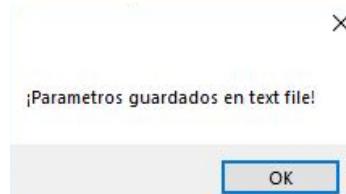


Figura 101: Parámetros guardados en text file

El cual archivo text file ha sido almacenado en la carpeta de recopilación de datos *Modo Generador*

### 11.3.2. Cargar archivos txt file

Cuando se requiera cargar un archivo txt file de configuración previa, se debe de presionar el botón *Cargar Ajustes* y se abrirá la siguiente ventana:

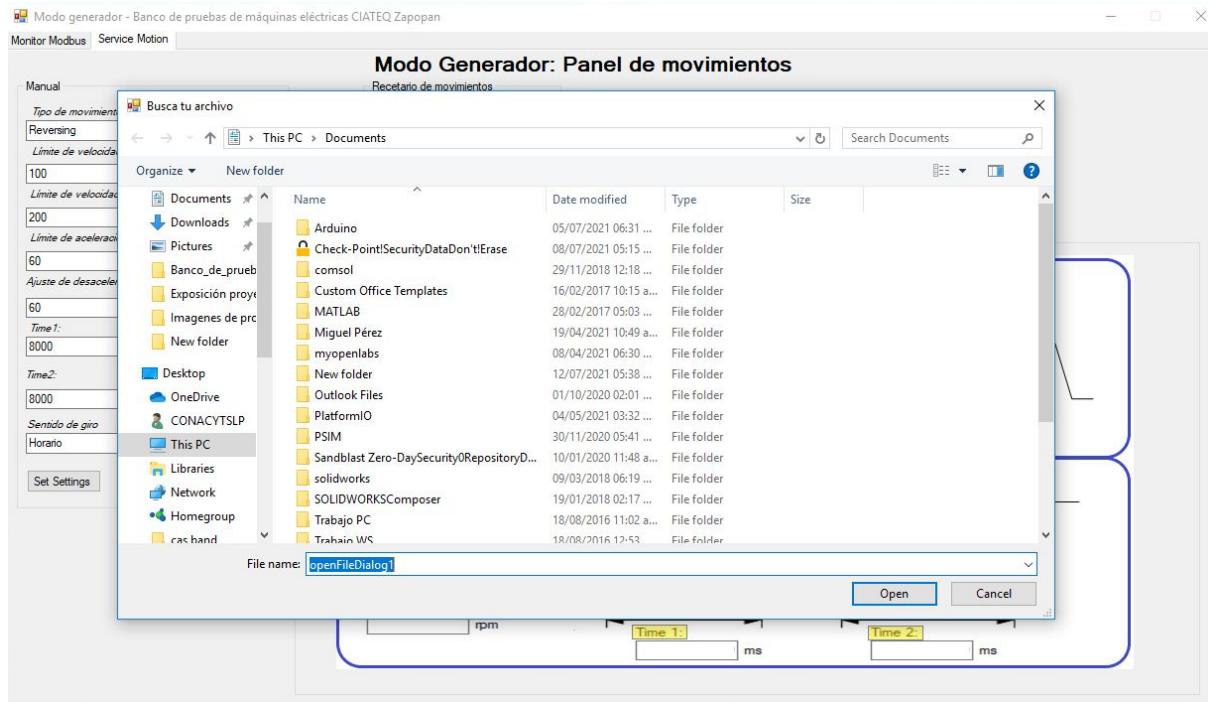


Figura 102: Ventana para buscar el archivo txt file y cargar ajustes

En esta ventana se debe de buscar el archivo txt file de los parámetros guardados anteriormente y presionar el botón *Open*, haciendo lo anterior el grupo *Manual* será llenado automáticamente de acuerdo al contenido del text file.

Únicamente el usuario deberá de presionar el botón *Set settings* para validar los campos de configuración y si todo es correcto configurar los registros del servodrive.

### 11.3.3. Pestaña “Monitor Modbus”

Una vez que se ingresaron los datos de configuración y se hayan enviado satisfactoriamente estos datos de configuración al sistema de control, se podrá acceder a la pestaña *Monitor Modbus*.

La ventana de esta pestaña es la siguiente:

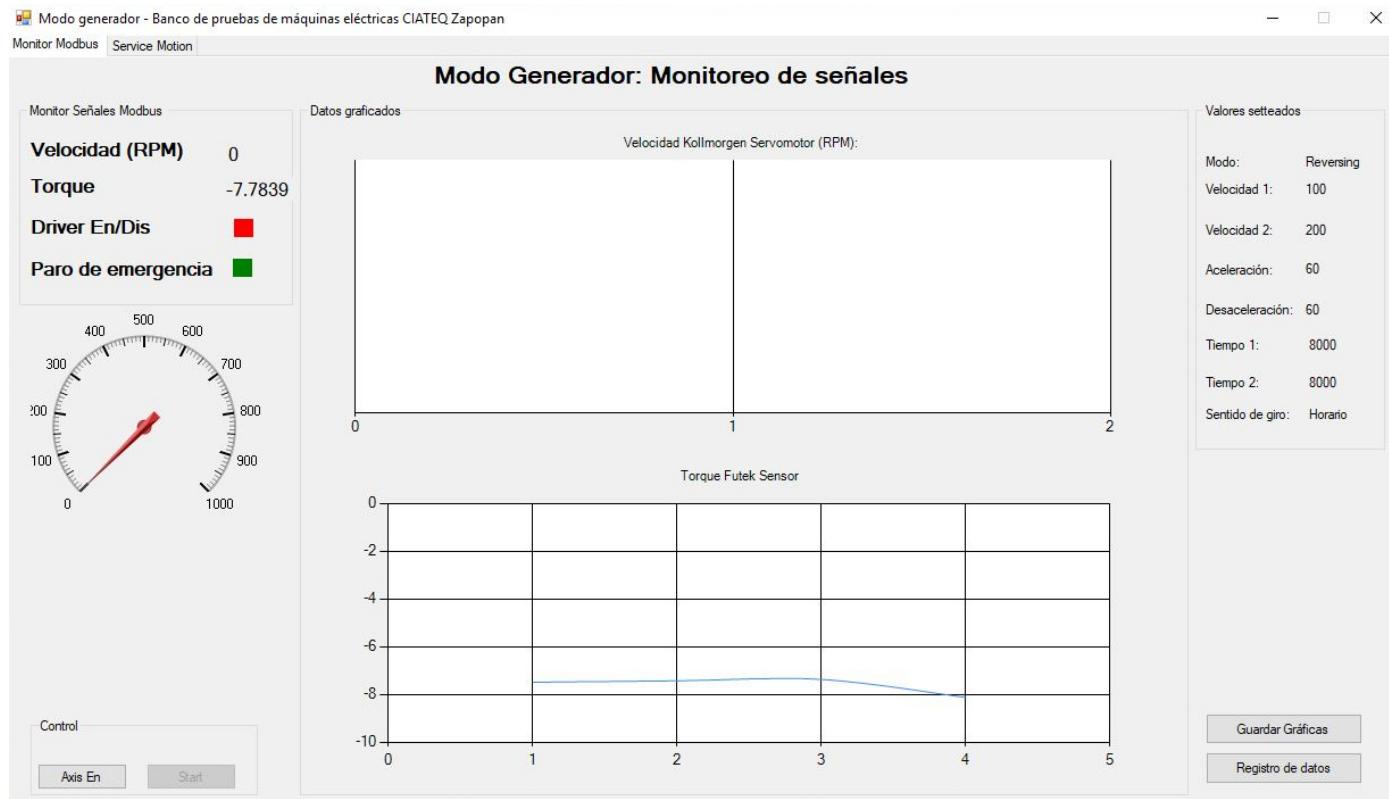


Figura 103: Ventana principal pestaña “Monitor Modbus”