DISEÑO E IMPLEMENTACIÓN DE UN BANCO DE PRUEBAS AUTOMÁTICO PARA LAS MORDAZAS MECÁNICAS EN ALLMATIC

José Chumán-Alvarado

Piura, noviembre de 2017

FACULTAD DE INGENIERÍA

Departamento de Ingeniería Mecánico-Eléctrica

Chumán, J. (2017). *Diseño e implementación de un banco de pruebas automático para las mordazas mecánicas en ALLMATIC* (Tesis de licenciatura en Ingeniería Mecánico-Eléctrica). Universidad de Piura, Facultad de Ingeniería. Programa Académico de Ingeniería Mecánico-Eléctrica. Piura, Perú.

U N I V E R S I D A D D E P I U R A

FACULTAD DE INGENIERÍA

Diseño e implementación de un banco de pruebas automático para las mordazas mecánicas en ALLMATIC

Tesis para optar el Título de
Ingeniero Mecánico-Eléctrico

José Francisco Chumán Alvarado

Asesor: Edilberto Horacio Vásquez Díaz

Piura, noviembre 2017

A Dios, mi familia y amigos

Prólogo

Este trabajo de titulación fue desarrollado en Alemania. La razón principal para esto es que Alemania es uno de los países en los que la ingeniería mecánica está más desarrollada. Cuando se recibió la oferta para llevar a cabo esta investigación, se encontró sumamente interesante la gran variedad de áreas que abarcaba el proyecto, tales como diseño mecánico, electrónica, automatización, etc., por lo que se aceptó dicha propuesta.

La importancia de este trabajo radica en que el módulo fabricado aumenta considerablemente la eficiencia del proceso de pruebas de los productos de la empresa, ya que la cantidad y calidad de los datos obtenidos es mayor, y el proceso es más rápido. Esto conlleva a que el proceso de diseño en general, sea también más rápido y, a su vez, más completo.

No puedo dejar de mencionar a aquellos quienes hicieron posible este trabajo: Mis padres, Enrique Chumán e Irene Alvarado, quienes me facilitaron realizar este viaje. Mi hermano mayor, Víctor Chumán, quien me acogió en su casa en Alemania y me ayudó en todo lo posible para adaptarme a la vida en ese país. Mi jefe y asesor en Alemania, Enrique Paiba, quien me guio durante todo el proceso de mi investigación. Y por último, a mi asesor en Piura, Edilberto Vásquez y a todas las otras personas que de alguna manera u otra me apoyaron en este proyecto.

Resumen

El presente trabajo consistió en diseñar un banco de pruebas capaz de probar todos los modelos de mordazas de la empresa ALLMATIC automáticamente; con el objetivo de mejorar la competitividad de la empresa, optimizando la proceso de realización de pruebas, y, por ende, el proceso de diseño.

Durante el diseño se asumió que las dimensiones y el comportamiento de algún eventual modelo futuro de mordaza no variaría lo suficiente de los actuales como para que el banco de pruebas no funcionara con dicho nuevo modelo. El módulo fue diseñado siguiendo las normas correspondientes de ergonomía. La estructura cuenta con piezas de fábrica y otras especialmente diseñadas para esta aplicación. Para la programación del motor, se contó con la ayuda de un asesor experimentado en el tema; y se usó el modelo de "máquina de estados".

Después de fabricado el módulo, se realizó pruebas con las mordazas para calibrar el banco de pruebas y, posteriormente, verificar su correcto funcionamiento.

En conclusión, al terminar este proyecto, el módulo fabricado quedó operativo; obtiene los datos requeridos de torque, fuerza y posición; además de ser mucho más rápido que los anteriores. Se implementó un control a distancia, y se añadió varios mecanismos de seguridad.

Índice

Introducción

La automatización ha estado presente en la industria desde hace mucho tiempo. Las ventajas de tener un sistema automático son varias, como por ejemplo, costos de producción reducidos, tiempos menores por ciclo, mejor utilización del espacio de trabajo, etc. Existen ejemplos de mecanismos de control con retroalimentación –o *feedback control* en inglés– desde 1620, con la invención del termostato [1]. Sin embargo, el control automático como se conoce hoy en día es resultado de la invención de las máquinas procesadoras de datos [2]. Con los recientes avances de estas últimas, el control automático ha experimentado un gran desarrollo, volviéndose más eficiente y económicamente cómodo. Como consecuencia de esto, cada vez más compañías implementan este tipo de control en sus sistemas. Sin embargo, no todos los métodos consiguen los mismos beneficios, y, por lo tanto, se vuelve necesario un diseño óptimo, teniendo en cuenta cada variable relevante del sistema o proceso inicial.

El presente proyecto consistirá en el diseño e implementación de un enfoque automático a un procedimiento de pruebas en la compañía alemana llamada ALLMATIC-Jakob Spannsysteme GmbH. En este documento se hará referencia a dicha compañía como "ALLMATIC".

ALLMATIC es una empresa que produce mordazas mecánicas de elevada presión. Una mordaza mecánica es un mecanismo que produce una fuerza de compresión en un elemento, asegurándolo para que se pueda realizar un trabajo sobre él (Wikipedia). Más detalles de este aparato serán explicados más adelante en este texto.

Como cualquier producto, estas mordazas deben pasar pruebas de calidad y desempeño antes de ser liberadas en el mercado. El comportamiento producido por la aplicación de un torque a la palanca del mecanismo es analizada en las pruebas. Actualmente, el procedimiento de aplicación de torque es automático, pero con una metodología *open loop*. La toma de datos es llevada a cabo manualmente, lo cual lleva a valores no muy precisos y a errores humanos. Además, la medición de la fuerza requiere que un sensor sea instalado cada vez que se lleva a cabo, y que este sea retirado después.

Para resolver el problema, una máquina automática para realizar las pruebas es propuesta. Este trabajo abarca el diseño, la construcción y la implementación de dicha máquina.

En el primer capítulo se introducirá el estado del arte en el tema. Los conceptos considerados como los más relevantes serán explicados. Conceptos generales serán considerados como conocidos por el lector. Durante la investigación no se encontró documentación de algún otro sistema parecido al que se implementará, así que este capítulo se centrará más en aquellos conceptos utilizados para el proyecto.

El segundo capítulo tratará del proceso de diseño en sí, tanto el diseño mecánico como el diseño del sistema. Se detallará las consideraciones y simplificaciones tomadas durante el diseño. Asimismo, se explicará brevemente los resultados de la verificación de los parámetros de seguridad. Se mostrará también el análisis modal de fallas y efectos.

El tercer capítulo abarca el proceso de construcción e implementación del banco de pruebas. Empezando por la parte mecánica, se mostrará de manera concisa las diversas etapas del montaje de la mesa y la estructura del motor. Luego se explicará la parte del control del sistema. Así mismo, se incluirá en este capítulo el manual de uso de la máquina.

Los resultados serán analizados en el cuarto capítulo. Se realizará también una breve comparación con los bancos de pruebas anteriores de la empresa.

En el último capítulo se describirá las conclusiones alcanzadas en este proyecto, así como las diversas posibilidades de mejora.

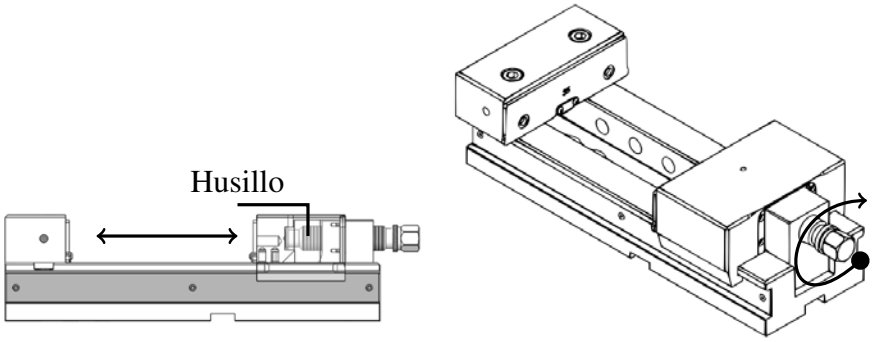
Capítulo 1
Estado del arte

1.1. Mordaza

Una mordaza es un mecanismo de sujeción. Aplicando una fuerza de compresión con dos bocas paralelas, se asegura firmemente una pieza para que se pueda realizar un trabajo sobre ella. En la mayoría de los casos, una de las bocas está fija y la otra es movible. El desplazamiento de esta última es obtenido mediante un sistema de husillo, en el cual un torque aplicado utilizando una palanca, en un extremo del husillo, resulta en un desplazamiento lineal y una fuerza en la dirección del eje del husillo. Las figuras 1a y 1b muestran dos vistas de una mordaza. En la primera se puede ver el sistema de husillo, y la segunda indica el punto de aplicación del torque.

1.1.1. Bocas

Las bocas son los elementos que presionan la pieza para sujetarla; esto significa que no debe haber movimiento entre las caras en contacto. Dicho movimiento es produ-



(a) Vista frontal (b) Vista isométrica

Figura 1. Mordaza
Fuente: Elaboración propia basada en la base de datos CAD de ALLMATIC

cido por fuerzas paralelas al plano definido por estas caras, las cuales son causadas por el método utilizado para trabajar la pieza, como por ejemplo, taladrado, fresado, etc. Para conseguir la condición de no movimiento, las fuerzas mencionadas deben ser menores que la fuerza de fricción, la cual es directamente proporcional al coeficiente de fricción, representado por la letra griega μ. Este coeficiente es determinado experimentalmente, no puede ser hallado mediante cálculos. Sin embargo, podemos decir que depende del material y la rugosidad de las superficies de los cuerpos en contacto. Normalmente, el valor de μ es mayor en superficies más ásperas. Por lo tanto, alterar la superficie del material puede generar una variación en el coeficiente de fricción.

Para esta aplicación, el objetivo es obtener una fuerza de fricción lo suficientemente alta para mantener a la pieza sujeta en su lugar. Normalmente, la modificación en el área de contacto, del objeto a sujetar, no es deseada; a veces, ni siquiera es posible. En objetos delicados, incluso la deformación causada por la fuerza de sujeción debe evitarse, y, en algunos casos, las bocas son cubiertas para la protección de la valiosa pieza. Estas restricciones nos dejan con la única posibilidad de modificar las bocas. La figura 2 muestra diferentes tipos de superficies.

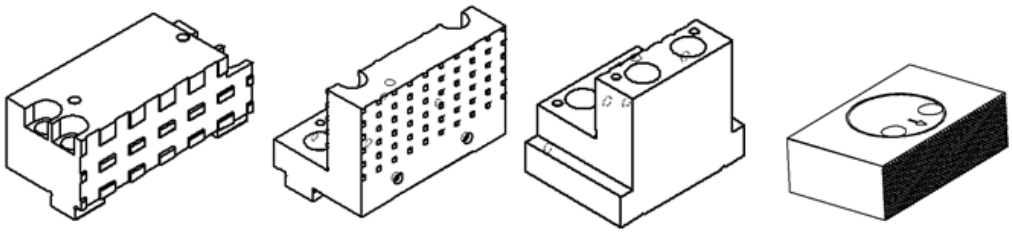


Figura 2. Bocas con diferentes superficies
Fuente: Elaboración propia basada en la base de datos CAD de ALLMATIC

1.1.2. Husillo

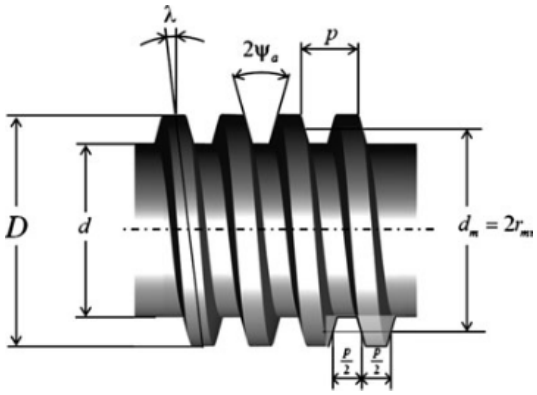
El husillo es un mecanismo que transforma movimiento rotatorio y torque a movimiento lineal y fuerza respectivamente. En el caso de las mordazas, esta fuerza resultante es lo que nos interesa, pues la fuerza de fricción será proporcional a ella. Las fuerzas en este tipo de sistema están determinadas por la fricción entre la tuerca y el husillo, y la geometría de la rosca del husillo. La nomenclatura de las magnitudes relevantes en una rosca son mostradas en la figura 3.

$$T = \pm\frac{F d_m}{2} \times \left(\frac{l + \pi\mu d_m \sec\alpha}{\pi d_m + \mu l \sec\alpha}\right) \tag{1.1}$$

Con la ecuación (1.1),se puede obtener el torque necesario para producir una fuerza deseada.

1.1.3. Multiplicador de fuerza

La mayoría de las mordazas disponibles comercialmente solo tienen el sistema de husillo para producir la fuerza de agarre. Lo que resalta de las mordazas de ALLMATIC es la adición de un multiplicador de fuerza –o *"Kraftverstärker"* en alemán–. El principio de funcionamiento de este amplificador es bastante simple. Usa una combinación de planos inclinados para convertir un desplazamiento inicial en otro más pequeño. Si consideramos la ley de conservación de la energía, podemos deducir que, para que esto



$$d_m = \frac{d+D}{2}$$

$\alpha = \psi_a$

$l = p \times$ número de entradas

$T =$ torque

$F =$ carga en la rosca

$\mu =$ coeficiente de fricción entre husillo y tuerca

Figura 3. Magnitudes en una rosca
Fuente: Friction-Induced Vibration in Lead Screw Drives, Vahid-Araghi, O. & Golnaraghi, F. (2011)

sea posible, la fuerza de salida del sistema tiene que ser mayor a la fuerza de entrada. La figura 4 muestra una vista de sección de uno de los tipos de multiplicador utilizados.

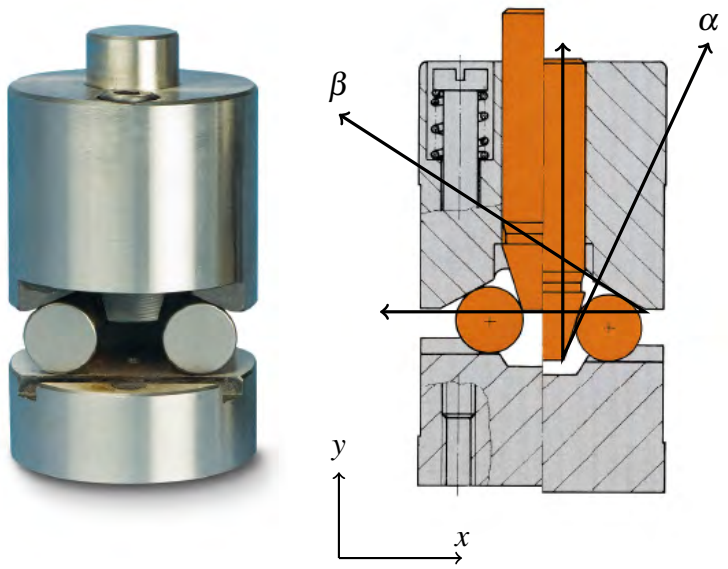


Figura 4. Kraftverstärker
Fuente: wwww.allmatic.de

Se aprecian 2 cilindros, 2 placas opuestas, y un perno separador. Considerando un diferencial de movimiento en el eje **y**, producido por el sistema de husillo, el cual contiene la inicial fuerza de compresión (F_1), se tiene un sistema cerrado con una energía de entrada igual a:

$$E_i = F_1 dy \tag{1.2}$$

Como resultado de este movimiento en el eje **y**, los cilindros se moverán en el eje **x**. La relación entre estos dos desplazamientos está determinada por el ángulo α:

$$x_1 = dy \cdot \tan\alpha \tag{1.3}$$

Existe también un ángulo en el punto de contacto entre los cilindros y las placas, el cual hace que los los cilindros empujen a las placas en la dirección del eje **y**. El nuevo desplazamiento en el eje **y** esta relacionado con el desplazamiento previo en la dirección del eje **x** por el ángulo β:

$$y_1 = x_1 \cdot \tan\beta \tag{1.4}$$

Entonces, el desplazamiento de salida sería la traslación expresada en (1.4). Utilizando la ley de conservación de la energía, y considerando condiciones ideales –sin fricción–, se obtiene la siguiente identidad:

$$F_1 dy = F_2 d\left(f\left(y\right)\right) \tag{1.5}$$

El segundo elemento en (1.5) representa la energía de salida. Combinando (1.3), (1.4) y (1.5) obtenemos, para este tipo de amplificador, una ventaja mecánica igual a:

$$\frac{F_2}{F_1} = \left(\tan\alpha \cdot \tan\beta\right)^{-1} \tag{1.6}$$

Utilizando los ángulos adecuados, se puede obtener la ventaja mecánica deseada. Considerando las pérdidas de fricción, desgaste de elementos, tolerancias y errores en la fabricación, etc., la ventaja real es menor a esta. Sin embargo, sigue siendo una gran mejora en comparación con aquellos modelos de mordaza que no tienen este elemento.

1.2. Servomotores

Tal como fue mencionado anteriormente, el objetivo de este proyecto es automatizar un proceso de pruebas. Para proveer el torque de entrada necesario, se utilizará servomotores; específicamente, un servomotor síncrono.

De acuerdo a Wikipedia [7], un servomotor, o servo, es un actuador rotatorio o lineal que permite el control preciso de la posición, ya sea angular o linear, velocidad y aceleración. En otras palabras, es un conjunto de un motor y sensores, combinación que permite un control de tipo *closed loop* o de lazo cerrado.

Un motor síncrono es un motor de corriente alterna en el cual la velocidad angular es igual a aquella de la corriente de entrada, en estado estacionario. Esta velocidad es conocida como “velocidad de sincronismo” y puede ser hallada con la expresión 1.7.

$$N_s = 120\,\frac{f}{p} \tag{1.7}$$

Donde :

$Ns =$ velocidad de sincronismo (RPM)

$f =$ frecuencia de la fuente (Hz)

$p =$ número de polos

1.2.1. Encoder

El elemento que convierte la posición rotacional en una señal de salida, analógica o digital se conoce como encoder o codificador (rotacional). Existe más de un tipo de encoder, como por ejemplo, conductivo, óptico o magnético. El primero de estos no se usa mucho en la actualidad, y el más usado es el óptico. Este último consiste en una fuente de luz, normalmente un LED; un detector de luz y un disco de código. El principio de funcionamiento es simple, el disco está construido de una forma que interrumpe la luz y forma un patrón, el cual es reconocido por el detector de luz [8]. De esta manera, la posición rotacional puede ser identificada.

Existen dos formas principales de codificar la posición del eje: codificación binaria y de Gray. Ambas usan el sistema de numeración binario; sin embargo, el código Gray tiene la característica de que dos valores sucesivos solo difieren en un bit. Un ejemplo de disco de Gray se muestra en la figura 5. Tomando cualquier par de números es posible comprobar que solo hay un bit de diferencia entre estos. Por ejemplo, entre el valor 7 y el 8, los cuales están representados por 00100 y 01100, solo el cuarto bit cambia su valor, de 0 a 1. La razón por la cual se prefiere este sistema es que los errores posibles son de menor magnitud [9]. Supongamos que trabajamos con los mismos valores, 7 y 8, en el sistema binario, es decir, 00111 y 01000. Si el cuarto bit es demasiado rápido y cambia antes de que los demás tengan oportunidad de hacerlo, se tiene un valor temporal de 01111, el cual representa el número 15 en binario. Trabajando con números mayores, el error se vuelve excesivo, si es que ya no lo era.

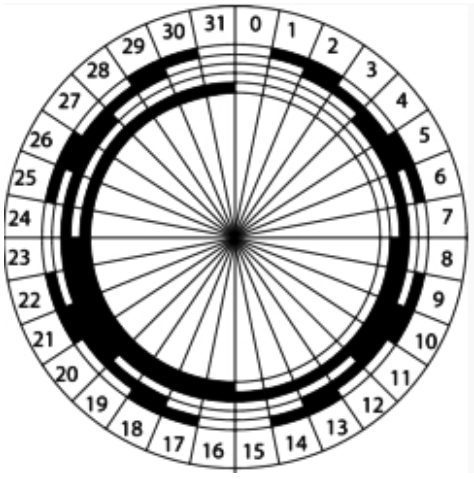


Figura 5. Disco de código
Fuente: catalogs.rockwellautomation.com

Otro criterio de clasificación divide a los encoder en encoder de vuelta simple y encoder de vuelta múltiple. En el caso de vuelta simple, se puede reconocer la posición del

eje pero solo en relación con la posición inicial del ciclo. Esto significa que no se puede saber cuántas vueltas ha dado el eje. Un encoder de vuelta múltiple es aquel que, además de lo que ofrece uno de vuelta simple, es capaz de indicar la cantidad de rotaciones hechas.

1.2.2. PWM

La velocidad del motor es determinada por la frecuencia de entrada. Por lo tanto, este valor debe ser regulable. Para conseguir una fuente de corriente alterna regulable, se usa convertidores AC/AC (corriente alterna a corriente alterna, por sus siglas en inglés), los cuales usan la modulación de ancho de pulso, o PWM por sus siglas en inglés.

Un transformador AC/AC básicamente rectifica una señal AC hacia una DC y luego, convierte esta última de nuevo a AC. La fase de inversión de este transformador lleva a cabo una PWM. Esta modulación consiste en la generación de una señal pulsante, la cual puede ser filtrada posteriormente para obtener una onda sinusoidal con la frecuencia deseada. La figura 6 muestra una señal pulsante con su señal sinusoidal correspondiente. Analíticamente, la señal seno puede ser obtenida con una descomposición de Fourier y aislando el primer armónico.

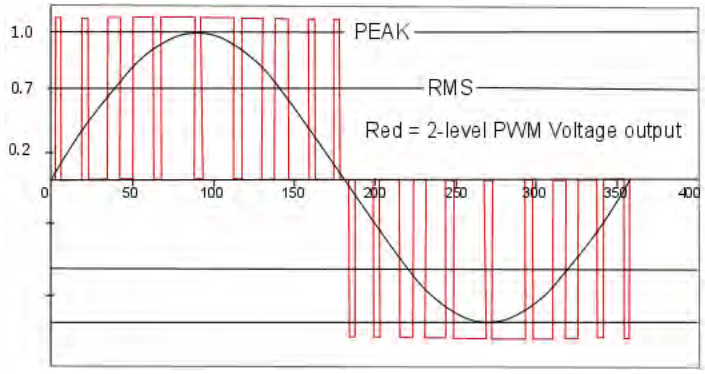


Figura 6. Onda sinusoidal desde una señal pulsante
Fuente:www.electricbike.com

1.3. Transductores

Un transductor es un dispositivo capaz de transformar una manifestación de energía de entrada, a otro tipo de energía de salida. Existen diversas formas de conseguir este fenómeno. Algunas de ellas, mostradas en [11], incluyen:

- Celda de carga con galgas extensiométricas
  - Galgas extensiométricas de resistencia eléctrica
  - Galgas extensiométricas de lámina
  - Galgas extensiométricas de semiconductor
  - Galgas extensiométricas de película delgada
  - Galgas extensiométricas de alambre
- Cristal piezoeléctrico
- Basadas en presión

  - Celda de carga hidráulica
  - Celda de carga neumática

- Otros
  - Dispositivos magneto-elásticos
  - Elementos vibratorios
  - Elementos dinámicos
  - Deformación plástica

La que recibe interés para este trabajo es la galga extensiométrica de lámina. A partir de ahora, cuando los términos "galga extensiométrica" o "*strain gauge*" en inglés, sean utilizados, se estará haciendo referencia a este determinado tipo.

Una galga extensiométrica es un sensor utilizado para determinar la deformación en un objeto. Normalmente estos objetos presentan una determinada forma geométrica y están diseñados para realizar mediciones de fuerzas. Sin embargo, una galga también puede ser adherida a un cuerpo de interés. A fin de obtener lecturas precisas, la galga debe ser correctamente adherida a la superficie, y dicha superficie debe estar tan limpia y ser tan regular como sea posible. En la figura 7b se muestra un ejemplo de una galga instalada. Una deformación en la dirección axial (eje horizontal en la figura 7a de una galga produce un cambio en su resistencia eléctrica. La relación entre este cambio y la deformación que la causó es una constante definida por el fabricante, llamada *Gauge Factor* o GF. Esta relación está expresada en (1.8).

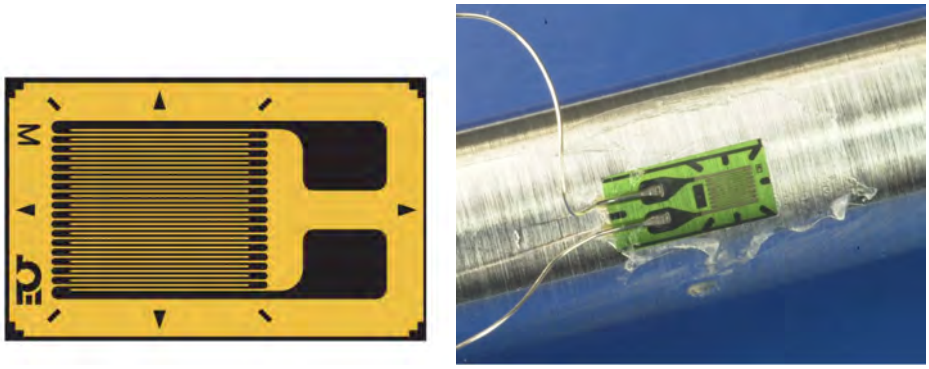
$$GF = \frac{\Delta R/R_g}{\varepsilon} \tag{1.8}$$

Donde :

$\Delta R =$ cambio en la resistencia

$R_g =$ resistencia de la galga no deformada

$\varepsilon =$ deformación



(a) Diseño básico de un *strain gauge*
Fuente: tacunasystems.com

(b) Galga instalada en un transductor
Fuente: www.doitpoms.ac.uk

Figura 7. *Strain gauge*

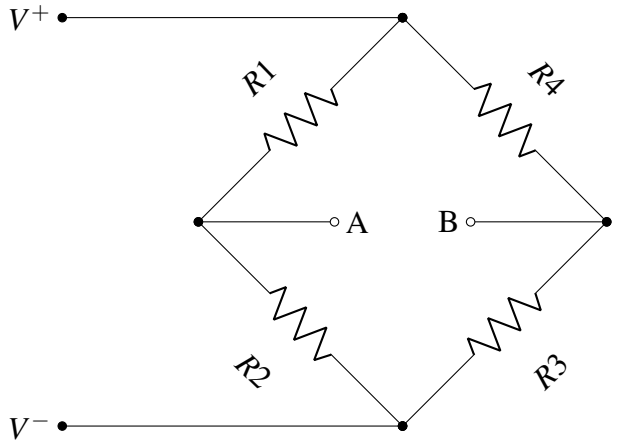


Figura 8. Puente Wheatstone
Fuente: Elaboración propia

La deformación se mide en partes por millón o micro-deformación (microstrain en inglés). [10] Los valores comunes de deformación están en el rango de 2 a 10000 microstrain. Con valores comerciales de alrededor de 2 para GF, se está en presencia de cambios de resistencia de aproximadamente 0.01 %. Para medir tal variación se vuelve necesario utilizar circuitos especiales. Entre estos, el más usado es aquel ilustrado en la figura 8, llamado puente Wheatstone. Algunas variaciones de este circuito se utilizan para reducir efectos no deseados, como la influencia de la resistencia de los conductores.

Un puente Wheatstone es un simple arreglo de resistores, con un voltaje aplicado V y una salida entre los nodos A y B. Cada resistencia está ubicada en un "brazo". Utilizando las leyes de Kirchhoff, la igualdad (1.9) es establecida.

$$V_{AB} = \frac{R_2R_4 - R_1R_3}{(R_1+R_2)(R_3+R_4)} \times V \tag{1.9}$$

De (1.9) se puede deducir que la salida será igual a cero, siempre que la relación entre la resistencia de dos brazos adyacentes sea igual. Esta condición de igualdad es conocida como puente "balanceado". Los puentes Wheatstone utilizados para medir *strain gauges* tienen siempre cuatro resistores idénticos. Dependiendo del tipo de puente, uno o más de los resistores son una galga extensiométrica. Un puente con solo una se conoce como cuarto de puente, uno con dos, medio puente, y uno con cuatro, puente completo. El cambio de resistencia es ínfimo, por lo que se puede despreciar las no-linearidades y decir que la variación del voltaje tendrá un comportamiento lineal tal como se describe en (1.10).

$$\Delta V = \frac{\partial V}{\partial R_1} \cdot \partial R_1 + \frac{\partial V}{\partial R_2} \cdot \partial R_2 + \frac{\partial V}{\partial R_3} \cdot \partial R_3 + \frac{\partial V}{\partial R_4} \cdot \partial R_4 \tag{1.10}$$

Combinando (1.9) y (1.10), y considerando un puente balanceado como condición inicial, obtenemos la expresión (1.11). Empleando (1.8), se llega a la expresión general para mediciones con galgas extensiométricas con un puente Wheatstone, descrita en (1.12).

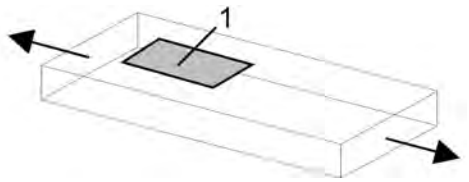
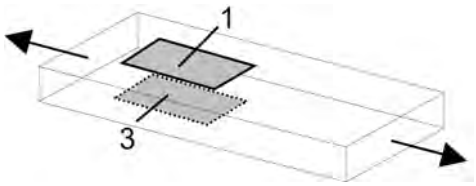
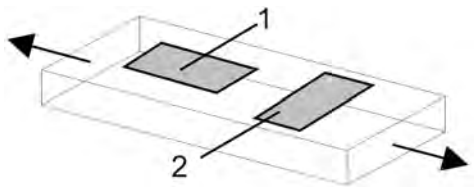
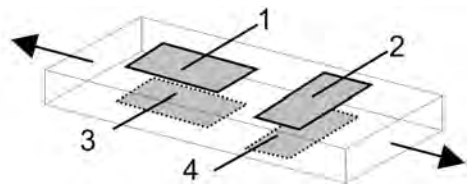
$$\Delta V = \frac{1}{4}\left(\frac{dR_1}{R} - \frac{dR_2}{R} + \frac{dR_3}{R} - \frac{dR_4}{R}\right) V \tag{1.11}$$

$$\frac{\Delta V}{V} = \frac{GF}{4}(\varepsilon_1 - \varepsilon_2 + \varepsilon_3 - \varepsilon_4) \tag{1.12}$$

Esta expresión es válida para todos los tipos de puente, siempre que, como condición inicial, el puente esté balanceado. En la práctica, no es fácil conseguir esto, dado que el circuito es sumamente sensible a cambios pequeños de resistencia. Usando un GF de 2, una deformación de 10 microstrain y una resistencia inicial de $120\,\Omega$, en (1.8), se obtiene una diferencia de resistencia de $2.4 \times 10^{-3}\,\Omega$. Entonces, si una de las resistencias tiene un valor que difiere del valor nominal en un par de mΩ, se obtendrá una lectura equivocada. Un balanceo del puente se vuelve necesario. En este proceso, resistores adicionales son añadidos, ya sea en serie o paralelo, para obtener un sistema balanceado.

La disposición de los *strain gauges* en la célula de carga también es importante, y depende de la fuerza a medirse y las otras fuerzas no deseadas cuya presencia se prevé como posible. Las tabla 1 contiene algunos arreglos comunes. Como es conocido, una deformación axial produce otra perpendicular a esta, relacionada con el módulo de Poisson υ. Conociendo esta relación, la suma de las deformaciones de cada brazo se puede expresar con (1.13) como un factor K multiplicado por una deformación.

Tabla 1: Configuración de los puentes para miembros uniaxiales

K	Configuración	Notas
1		Debe utilizarse una galga de corrección o compensación en un brazo adyacente (2 ó 4) para compensar los efectos térmicos.
2		Rechaza deformaciones por flexión pero no presenta compensación de efectos térmicos; requiere la adición de galgas de corrección en los brazos 2 y 4.
$(1+\upsilon)$		Efectos térmicos compensados pero sensible a deformaciones por flexión
$2(1+\upsilon)$		Mejor configuración: compensación de efectos térmicos y rechaza deformaciones por flexión

Fuente: AE3145 Resistance Strain Gage Circuits

$$\frac{\Delta V}{V} = K \times \frac{GF}{4}\varepsilon \tag{1.13}$$

Una práctica común consiste en instalar más de una galga en la misma dirección, de manera que sean afectadas por el mismo tipo de deformación, y conectarlas en serie formando un brazo. Si bien esto no incrementa la sensitividad del puente, ayuda a promediar las lecturas de deformación. Este comportamiento se puede probar fácilmente tomando un término de (1.11). Si se tiene 'n' galgas con una resistencia igual a R, la resistencia total por brazo (R_a) será $n \cdot R$, y:

$$\frac{dR_{a1}}{R_{a1}} = \frac{dR_a^I + dR_a^{II}}{nR} \cdots = \frac{\varepsilon_I + \varepsilon_{II}}{n} \cdots \times \frac{1}{R} = \frac{\varepsilon_{promedio}}{R} \tag{1.14}$$

En el caso de elementos sujetos a torsión, adicionalmente a la rotación en la dirección de las galgas explicada en la tabla 2, se realiza una pequeña reducción del diámetro de la célula de carga, en la zona donde las galgas serán instaladas. Con esto se consigue generar localmente una deformación mayor, lo cual incrementa la sensitividad del sistema de medición. Esta reducción de sección está ilustrada en la figura 9, la cual muestra también algunos componentes de un sensor de torque común. Entre estos, vale la pena resaltar el transformador rotativo. Este elemento permite conectar el sistema de *strain gauges* con el resto de la electrónica sin preocuparse por los problemas de fricción que conlleva la utilización de anillos rotatorios. Otro aspecto a considerar es que, en estos sensores, existe un lado de medición y otro de accionamiento o trabajo. Una conexión errónea conducirá a medidas incorrectas y a una posible falla del sensor.

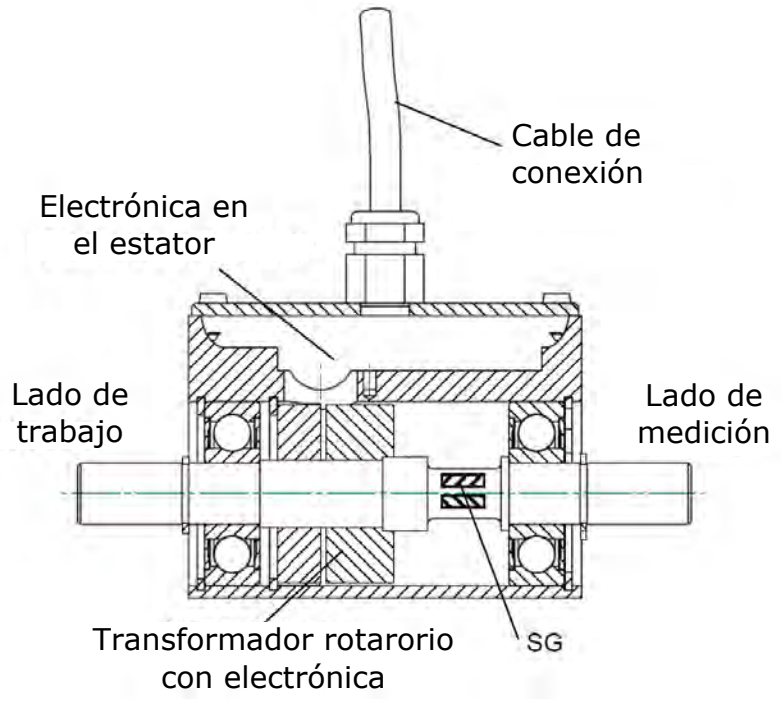
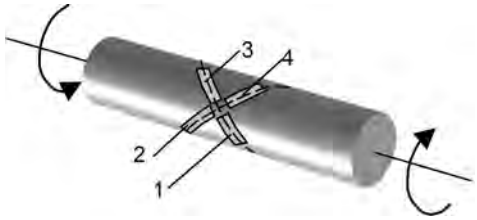


Figura 9. Vista de sección de un sensor de torque
Fuente: Manual de operación de un sensor de torque de Lorenz Messtechnik

Tabla 2: Configuración de puente para miembros de torsión

K	Configuración	Notas
2		Medio puente: Galgas a $\pm 45°$ de la linea central detectan las deformaciones principales, las cuales son iguales y opuestas para torsión pura; fuerzas de flexión o fuerzas axiales inducen deformaciones iguales y son, por ende, rechazadas; los brazos presentan compensación de efectos térmicos.
4		Mejor configuración: versión en puente completo del caso anterior; rechaza deformación axial y de flexión y tiene compensación de efectos térmicos.

Fuente: AE3145 Resistance Strain Gage Circuits

Capítulo 2
Diseño

Antes de diseñar el sistema, es necesario saber exactamente qué es lo que se tiene que realizar. El objetivo de este trabajo es diseñar un sistema para automatizar el proceso de probar las mordazas. Para estas pruebas, la mordaza está fija a una mesa de trabajo. Un motor eléctrico provee el torque necesario para que las bocas empiecen a cerrarse. Un sensor de posición indica cuándo debe detenerse el motor. Esta posición es mantenida, sin aplicación de torque, por un tiempo determinado, después del cual el motor empieza a girar en sentido contrario para abrir la mordaza. Este ciclo es repetido tantas veces como sea indicado por el encargado de llevar a cabo las pruebas.

La idea de este procedimiento es determinar qué tan rápido se deteriora una mordaza. Después de que el número de ciclos establecido se ha completado, la fuerza de compresión que produce la mordaza se mide con un sensor de fuerza y se anota el valor obtenido; esto es, la toma de datos es manual. Con suficientes medidas, es posible estimar el comportamiento del producto relacionado con el tiempo de uso.

Aun cuando este procedimiento funciona, es en general muy simple, y está muy lejos de ser óptimo. Por ejemplo, el acople entre motor y mordaza es de tal forma que las pérdidas de trabajo son muy altas. No obstante, tiene una ventaja en cuanto a que se puede utilizar con todos los modelos de mordazas.

Otro aspecto que se podría mejorar es que el único parámetro medido es la fuerza. Esta magnitud es la salida del sistema, y es lo que determina el estado de la mordaza. Sin embargo, existen otros aspectos que influyen dicho estado. Entre estos, el más relevante es el torque de entrada. La relación entre entrada y salida es fundamental para determinar el desempeño de la mordaza. En el capítulo anterior se explicó algunas ecuaciones que definen esta relación; con ellas se puede calcular el comportamiento ideal del sistema. Cuando los resultados reales difieren excesivamente de lo ideal, se puede decir que ya se excedió el tiempo de vida, que hace falta realizar mantenimiento, o que hubo un error en la construcción.

El método para controlar la apertura y el cierre de la mordaza está definido por un tiempo y un sensor de posición. Esto indica que el sistema es de lazo abierto. Además, la recolección de datos está limitada por el hecho de ser realizada manualmente, y los resultados no permiten obtener conclusiones precisas.

Lo que es requerido para este proyecto es entonces, un sistema automático que sea capaz de resolver estos problemas y, a su vez, mejorar la eficiencia del proceso.

Una vez que los objetivos han sido establecidos, todavía quedan unos aspectos que todo diseñador debe considerar. Por ejemplo, en un proceso de diseño siempre hay que tener en cuenta que, en la realidad, los detalles no serán exactamente como fueron pensados o dibujados. Ningún proceso de producción es perfecto, y siempre habrá una desviación entre las dimensiones de diseño y las reales. Esto nos lleva entonces a introducir el concepto de tolerancias y ajustes.

Estos conceptos, a pesar de ser similares, no deben confundirse. Una tolerancia es el máximo error no intencional que se puede permitir en una dimensión. Esto quiere decir que el valor de la dimensión ahora no es único sino que puede estar en un rango de valores dentro de los cuales la pieza puede desempeñar su función correctamente. Se podría decir que, mientras más pequeño sea este rango, más exacta será la dimensión y, por ende, la pieza será de mejor calidad. Sin embargo, establecer tolerancias extremadamente ajustadas, implicaría elevar considerablemente el costo de la pieza y el tiempo de fabricación. Como Glassen (2013) dice, las tolerancias deben ser tan generosas como sea posible en la fase de diseño para mantener bajos los costos, a menos que sean requeridos para un encaje o función apropiados de la pieza/ensamblaje.

Como consecuencia, las tolerancias deben indicarse para las dimensiones más importantes. Para dimensiones sin tolerancias suele usarse valores predefinidos. En el caso de piezas estándar como pernos, roscas, rodamientos; existen normas DIN que determinan los valores a utilizar. En piezas particulares, el fabricante establece las tolerancias que necesitan para ser ensamblados. Asimismo, los valores utilizados por el fabricante son especificados en el catálogo o la documentación del producto para posibilitar el diseño de un sistema que incluya dicho producto. Para otras medidas, un análisis previo de la función de la pieza es necesario para determinar el tipo óptimo de tolerancia. En este último caso se necesita cierto nivel de experiencia en este campo, y para la mayoría de decisiones para el presente proyecto conté con una gran ayuda por parte de mi co-asesor.

Un ajuste es el máximo error intencional que se puede permitir. Este error intencional es útil para ciertos casos. Por ejemplo, se puede tomar el caso del acople entre un eje y un agujero. Como ya se explicó, las medidas reales están entre un rango de valores. Si, al dimensionar estos dos elementos, se ha especificado el mismo diámetro, existen dos extremos posibles, en los cuales uno de los elementos tiene una dimensión en el extremo superior del rango, y, el otro, en el extremo inferior. Si es el eje el que está en el extremo superior, se produce un acoplamiento ajustado o con interferencia. Si el eje está en el extremo inferior, se produce un acople con juego. Uno no puede arriesgarse a tener una u otra situación, pues normalmente, en el diseño, es solo una de ellas la que se desea. Entonces, una pequeña desviación en la dimensión es introducida para asegurarse de que solo existe una posibilidad. También es posible garantizar esto utilizando tolerancias; sin embargo, los ajustes permiten visualizar rápidamente el tipo de acople que se necesita.

El proceso de construcción o ensamblaje también debe ser considerado. Incluso con un dimensionamiento correcto de las partes y un proceso de manufactura excelente, el ensamblaje puede tornarse complicado. Un buen diseño, por ejemplo, debe dejar suficiente espacio para que el constructor pueda utilizar sus herramientas cómodamente. Formas y uniones muy complicadas deben ser evitadas a menos que sean realmente necesarias. En los puntos de ensamblaje o unión se debe incluir un pequeño chaflán para facilitar el montaje.

2.1. Mesa de trabajo

Para las pruebas se requiere un torque máximo de 200 Nm. El valor del torque debe ser conocido en todo momento, para poder monitorear el proceso. Tal torque no puede ser conseguido con un simple motor eléctrico, por lo que se consideró necesario añadir una reducción. Hasta ahora se ha definido 3 componentes: un motor, con todos los accesorios necesarios; una reducción y un sensor de torque.

La mesa debe ser capaz de soportar el peso de la mordaza que está siendo probada y todos los otros componentes. La mordaza más pesada producida por ALLMATIC pesa alrededor de 60 kg. Asumiendo un peso combinado de 30kg entre motor y reducción, y 25 kg en accesorios, la mesa necesita resistir aproximadamente 1200 N.

A pesar de ser automático, el módulo será operado por personas. Como consecuencia, la ergonomía del sistema debe ser tomada en cuenta, y, como en la mayoría de casos, existen normas en esta área.

El operador trabajará parado. La altura es un parámetro que puede cambiar drásticamente de persona a persona, por consiguiente, la altura de la mesa debe cambiar también. Para conseguir una altura dinámica que se pueda adaptar a los diversos usuarios, se utilizó columnas elevadoras. Según la norma correspondiente, DIN_EN 527-1:2011, los rangos de alturas según los diversos casos son aquellos especificados en la tabla 3. Este estándar recomienda utilizar los tipos A y B.

Tabla 3: Altura de las mesas según DIN_EN 527-1:2011

Tipo	Área de aplicación		
	Sentado	Parado	Mesa sentado-parado
A[a] y B[b]	650 a 850	950 a 1250	650 a 1250
C[c]	680 a 760	1000 a 1180	680 a 1180
D[d]	740 +/- 20	1050 +/- 20	

Todas las medidas están en mm

[a] Tipo A: Mesas de altura ajustable con un rango de ajuste amplio (altura ajustable=la altura puede variar durante el uso)

[b] Tipo B: Mesas de altura ajustable con un rango de ajuste amplio (altura ajustable=la altura puede ser adaptada al tamaño corporal del usuario cuando la mesa es instalada

[c] Tipo C: Mesas de altura rígida

[d] Tipo D: Mesas de altura ajustable con un rango de ajuste restringido

Fuente: http://www.buero-forum.de/index.php?id=740

Las columnas elevadoras elegidas para la mesa son aquellas de la compañía Rose + Krieger, y se llaman RK Powerlift. Las características de dichos dispositivos, necesarias para el diseño estructural, son la carga y el torque que deben soportar, los cuales son 2000 N y 125 N m –dinámico– y 250 N m –estático– respectivamente.

La estimación inicial de la carga fue de 1200 N, así que este modelo debería ser suficiente. Sin embargo, esta no es la única carga que van a soportar las columnas; una parte de la estructura de la mesa estará encima de ellas. Con unos cuantos cálculos se puede establecer en 2800 N.

Todos estos valores fueron estimados considerando un sistema estático. Los efectos dinámicos son considerados despreciables debido a las bajas velocidades con las que se

trabajará. La posibilidad de una fuerza externa extraordinaria está contemplada en un factor de seguridad para los dispositivos elevadores, el cual será fijado a un valor tentativo de 1.5.

La figura 10 muestra la columna elevadora utilizada par la mesa. Tiene un par de ranuras en dos de sus lados opuestos para permitir su acople con otros elementos. También presenta agujeros roscados M10 para asegurar las partes superior e inferior con la estructura deseada. A causa de la especificidad de la aplicación, ninguno de los elementos de unión proveídos por el fabricante eran adecuados; por consiguiente, se diseñó elementos personalizados para la situación.

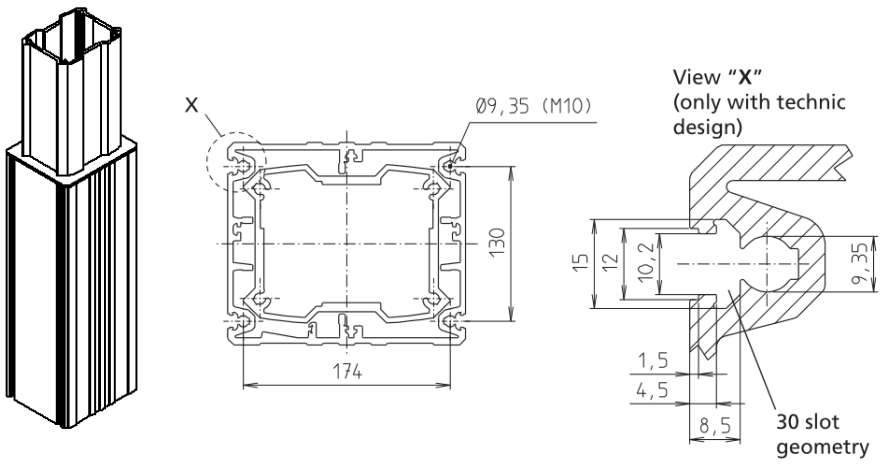


Figura 10. RK Powerlift
Fuente: Catálogo de productos de Rose und Krieger

El cuerpo principal de la mesa está compuesto por perfiles de aluminio, cuya respectiva área de sección se muestra en la figura 11, y provienen del mismo fabricante que las columnas elevadoras. Este tipo de perfiles esta diseñado para soportar grandes cargas. Como lo indica la imagen, los perfiles también tienen ranuras para su acople con otros elementos, utilizando tuercas deslizantes según la norma DIN 508. También existe la opción de utilizar los conectores del mismo proveedor. Se utilizó una combinación de los últimos y tuercas deslizantes.

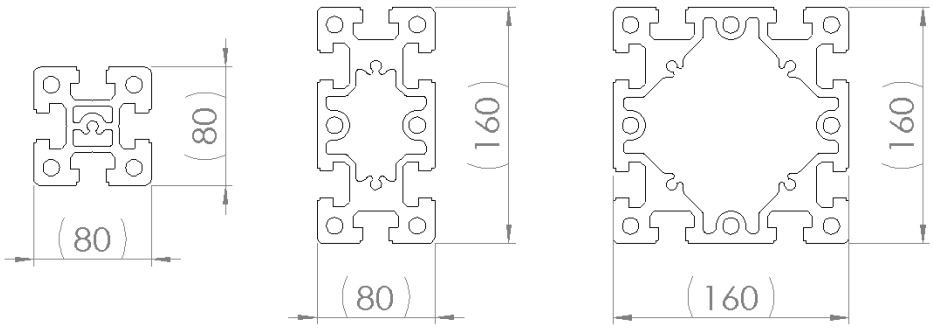


Figura 11. Perfiles de alto rendimiento
Fuente: Datos CAD de Rose und Krieger[1]

[1]Todas las dimensiones en las figuras están en milímetros a menos que se indique lo contrario

Para la parte superior, el primer diseño contemplaba una simple estructura rectangular con vigas de refuerzo a lo largo. La parte inferior consistía en dos vigas como base para las columnas, una viga uniendo estas bases para darle estabilidad, y una extensión de la linea descrita por esta última viga, con el objetivo de proporcionar una base para el armario que contendrá la electrónica del sistema. Estas dos estructuras iniciales están ilustradas esquemáticamente en la figura 12. Ambas partes serían conectadas con las columnas, cuya altura puede ser controlada para obtener el rango de altura recomendado, esto es, desde 950 mm hasta 1250 mm. Desde el punto de vista de ser capaces de resistir la carga esperada, este diseño es más que suficiente. Sin embargo, no todas las mordazas tienen las mismas dimensiones, así que es necesario implementar una forma de posicionarlas de manera que sus ejes estén alineados con el del motor. Como primera propuesta se pensó en fabricar distintas placas como base de cada mordaza, las cuales serían unidas con tuercas deslizantes. De esta forma, cada mordaza estaría a la altura adecuada. Esta idea fue descartada debido al gran número de placas que serían necesarias y a que resultaría demasiado tedioso realizar el cambio de placas para cada modelo.

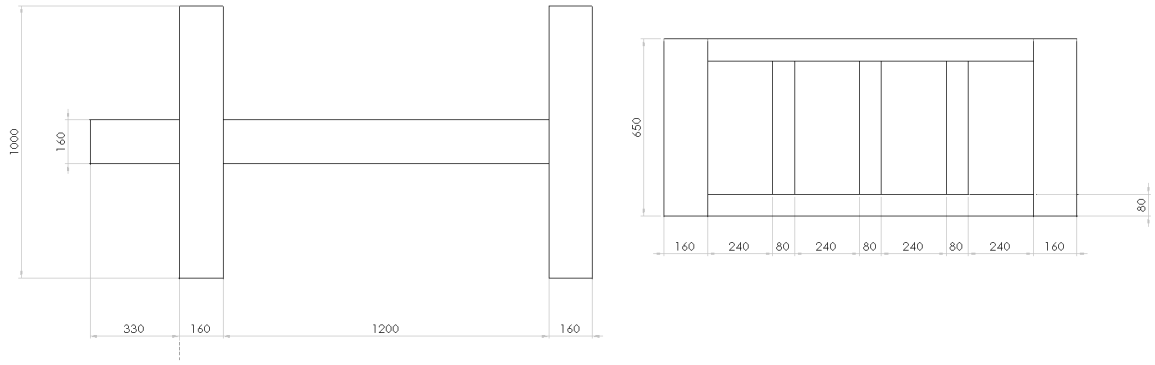


Figura 12. Primer diseño
Fuente: Elaboración propia

Ante la necesidad de una solución más eficiente, se planteó un mecanismo consistente en una placa o plato metálico cuya posición vertical pueda variar para adaptarse a la mordaza que se quiere usar, unido a otro de posición fija, el cual servirá como base para el mecanismo. Este elemento, mostrado en la figura 13, también debe tener agujeros roscados que permitan fijar las mordazas con pernos y elementos de sujeción. Para soportar la sección, un par de perfiles fueron añadidos a la sección superior de la mesa. El aparato utilizado para generar la variación de altura fue un tornillo de rosca trapezoidal. Para el alineamiento de las dos placas, se instaló 4 guías, con frenos en dos de ellas para fijar la posición una vez que fuera determinada como óptima. Estos frenos también sirven como precaución pues existe el riesgo de que el tornillo, a pesar de ser de bloqueo automático, pierda dicho bloqueo ante una carga elevada. En la figura 13 también se aprecia un canal atravesando el plato superior; este sirve para centrar las mordazas, pues la gran mayoría de estas posee una ranura para introducir un elemento para el centrado, y, con este canal definiendo el centro, se puede obtener una posición precisa. El rango de posibles alturas escogido es mayor al rango de alturas necesarias para las mordazas existentes; esto se hizo contemplando la posibilidad de nuevos modelos más pequeños o grandes.

Combinando todos los elementos descritos, y con unas pequeñas modificaciones, el diseño mostrado en la figura 14 es alcanzado. Los parámetros de diseño fueron revisados para corroborar que la mayoría de modos de fallo hayan sido considerados; su posibilidad

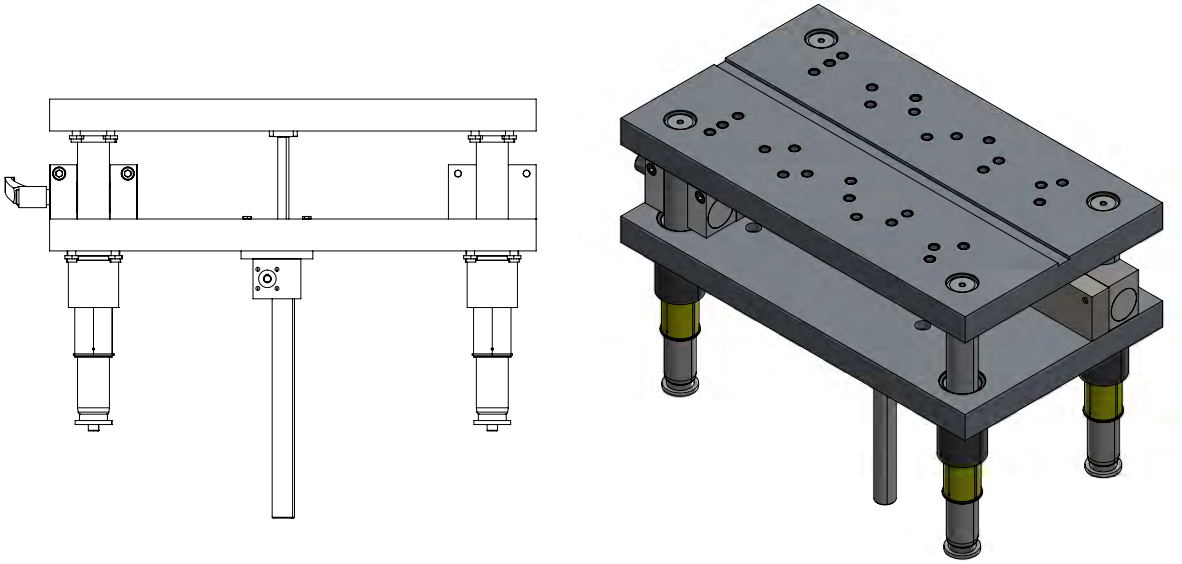


Figura 13. Plato de altura regulable
Fuente: Elaboración propia

haya sido reducida y, en la medida de lo posible, hayan sido eliminados. Este proceso se explica a continuación.

En cuanto a las columnas elevadoras, no se puede hacer modificaciones en sus parámetros pues estos son definidos por el fabricante; lo único que se puede hacer es elegir entre modelos. Una vez elegido el modelo, el diseño tendrá que ser realizado en relación con este.

Conociendo el peso de la mayoría de los elementos de la mesa, el peso que las columnas deben soportar es calculado en aproximadamente 2800 N. Ya que el máximo por columna es 2000 N, se tiene un factor de seguridad (FS) de aprox. 1.4, lo cual está suficientemente cerca del valor al que se quería llegar, 1.5. También se puede calcular un FS para el torque. Como se mencionó anteriormente, el sistema es considerado como estático, por lo que se trabajará con el límite de 250 N m por columna. Con un torque de diseño máximo de 200 N m, el FS para este parámetro es de 2.5, lo que indica que la estructura debería ser capaz de soportar torques extraordinarios de ser necesario.

Para el resto de la estructura, el sistema puede considerarse como una combinación de elementos estructurales. Una vez que fue comprobado que estos últimos no fallan, se puede analizar los elementos de conexión. Dichos elementos tienen varios límites de momento y fuerza, dependiendo de la dirección en la que se apliquen estas cargas. Considerando los valores más bajos, los cuales son 2900 N y 560 N m, se puede deducir que ninguno de estos conectores fallará. Sin embargo, un análisis en los puntos más críticos tiene que realizarse para verificar que estos límites no se excedan. El diseño de la figura 13 es un poco más complicado. En el catálogo de FIBRO, el proveedor escogido para las guías, existe una matriz diseñada para facilitar la elección de columnas y casquillos. Para una configuración como esta, de 4 pilares, es requerida una tolerancia relativamente holgada. Las dimensiones de los pilares son 40 mm de diámetro y 355 mmde largo. Las cargas que afectan directamente a las columnas son el peso del plato superior y las mordazas, y el torque de las pruebas. Estos pesos también generan un momento. Aparte de eso, existe la posibilidad de alguna fuerza

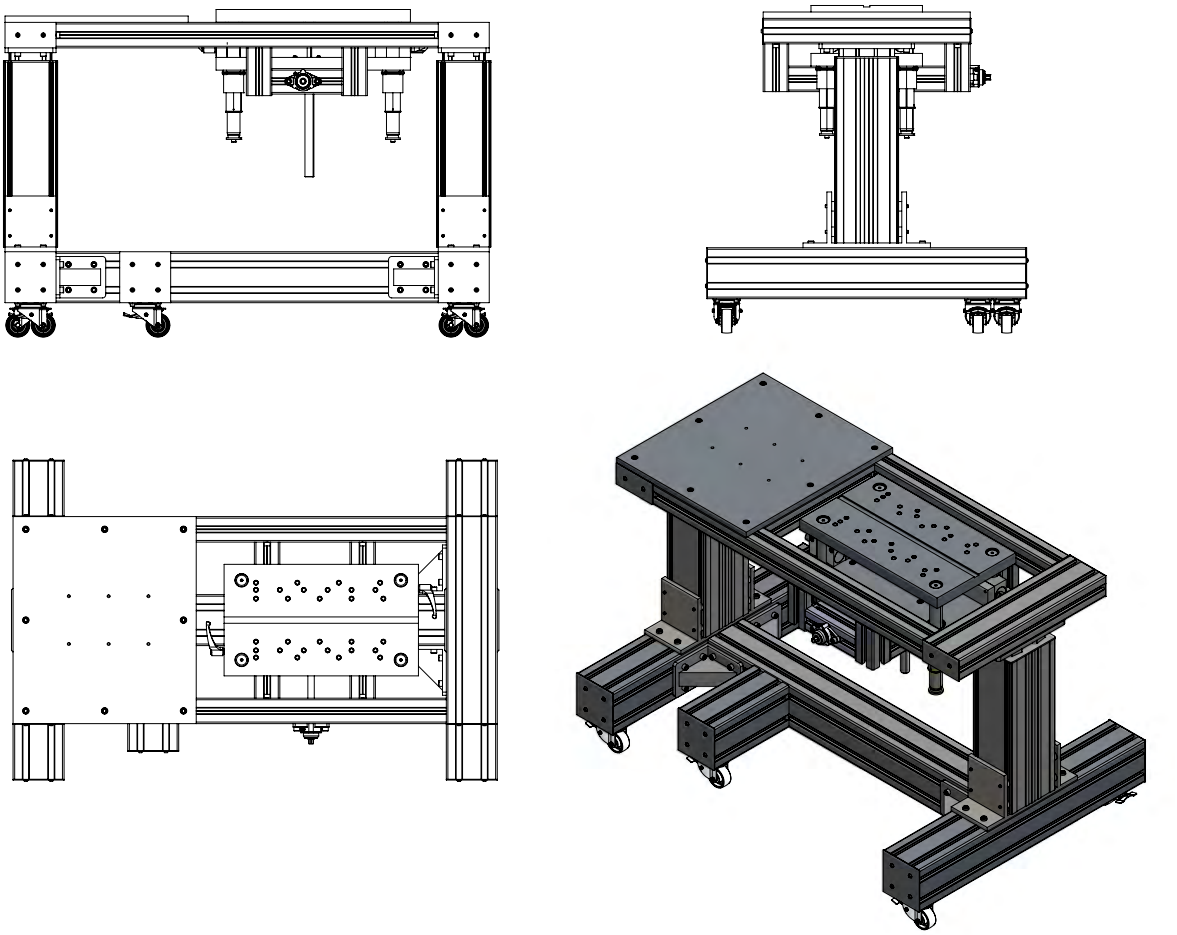


Figura 14. Diseño final de la mesa
Fuente: Elaboración propia

exterior extraordinaria, por ejemplo, si alguien se apoya en la mesa o la empuja. Con estos datos y el material, se puede calcular la carga máxima que pueden soportar; la cual resultó mayor a la necesaria.

Para el sistema de rosca trapezoidal, el modelo escogido fue el más pequeño que el proveedor ofrece. Sin embargo, es capaz de levantar hasta 2.5 kN, lo cual es más que suficiente. El sistema debe estar situado lo más cerca posible al centro de la placa inferior para obtener la mejor eficiencia. Sin embargo, con la estructura que lo rodea es casi imposible alcanzar el punto de aplicación del torque con el instrumento adecuado; por lo que fue necesario añadir una extensión, la cual fue simplemente un cilindro con un agujero en un extremo, que encaja en el punto previamente inalcanzable, y una cabeza hexagonal en el otro extremo.

2.1.1. Cubierta

El comportamiento del sistema involucra un movimiento de cierre. Para las pruebas, existe la posibilidad de trabajar con fuerzas mayores a las nominales. Incluso sin este exceso de fuerza, la salida de las mordazas se encuentra alrededor de 80 kN. Una fuerza de esta magnitud puede causar daños físicos muy graves en caso de accidente, así que nadie debe ser capaz de acercarse a las superficies de compresión durante la operación del sistema.

Como medida de protección, la velocidad de cierre es establecida lo suficientemente baja para que cualquiera tenga tiempo de reaccionar ante una situación de riesgo de accidentes. Sin embargo, otros riesgos están presentes. El riesgo más grave, y la principal razón por la cual es necesaria una cubierta, es que las bocas pueden romperse. Cuando se diseña nuevos modelos de mordaza, además de preocuparse por conseguir una fuerza mayor, es necesario diseñar bocas que sean capaces de soportar tal fuerza. El valor máximo de carga que pueden soportar es de interés, pues es con él que se puede determinar el factor de seguridad. Para hallar este valor, es necesario llegar a romper la boca.

Mientras mayor sea la fuerza que resisten las bocas, más peligrosa será su ruptura. Esto se debe a que casi toda la energía almacenada en forma de deformación se transforma en energía cinética. Un pedazo de metal, con superficies toscas debidas al proceso de ruptura, y a grandes velocidades, dañaría gravemente a cualquiera que golpee. La cubierta debe evitar esta colisión.

La cubierta contará también con un interruptor de seguridad, el cual no le permitirá ser abierta a menos que las condiciones del sistema sean determinadas como seguras por el sistema.

2.2. Motor

La parte más importante del banco de pruebas es el actuador, en este caso, el motor. El motor fue elegido considerando la máxima carga que debe suministrar, esto es, 200 N m. La aplicación de este momento no será continua. El torque se aplica hasta que la mordaza alcanza la fuerza de compresión deseada, y, cuando este punto ha sido alcanzado, la misma configuración mecánica de la mordaza mantiene la posición hasta que un torque en sentido contrario es aplicado. Así que no es tan crítico si el motor está al límite al ejercer este momento. Sin embargo, para cualquier máquina, no es una buena práctica exceder el límite de trabajo, ya que esto lleva a una reducción del tiempo de vida útil y posibles fallos.

El motor elegido fue de la compañía Stöber Antriebstechnik. La tabla 4 enumera sus características más importantes. Este motor cuenta con una reducción de engranajes cónicos integrada, lo cual le permite conseguir un torque de esta magnitud. Con un valor de 191 N m, se estará trabajando debajo del límite la mayor parte del tiempo, excepto cuando se necesita los 200 N m. Sin embargo, hay que tener en cuenta que este límite es el del trabajo continuo. Como ya se indicó, el motor no va a trabajar continuamente, por lo que el límite es aquel que se observa en las gráficas del motor, mostradas en la figura 15.

Otro parámetro a considerar en el diseño de motores es el torque de aceleración. Adicionalmente al torque de trabajo requerido, es necesario vencer la inercia para poder variar la velocidad del sistema, y, por consiguiente, generar un movimiento. La inercia total del sistema es calculada.

Para hallar este valor, se debe calcular la inercia para cada elemento. La inercia del motor es indicada en las especificaciones del fabricante. Los otros valores deben ser hallados con las fórmulas correspondientes. Como ya se explicó, el principio de funcionamiento de una mordaza es el de husillo. De acuerdo con [12], la inercia total de un sistema de husillo, con un motor como actuador, se puede calcular con la ecuación (2.1). Aquí, el sistema de husillo es considerado como un cilindro, cuya fórmula para la inercia es el segundo elemento de la ecuación. Es importante tener en cuenta que la reducción no solo transforma el torque sino también la inercia. En este caso, el valor suministrado es el de la salida de la reducción,

Tabla 4: Parámetros del motor

Parámetro	Valor
Voltaje	540 V
Factor-KE	119 Vmin/1000
Potencia	3.05 kW
Velocidad	3000 1/min
Torque (M_1)	9.70 Nm
Corriente nominal	6.90 A
Velocidad nominal (n_1)	3000 1/min
Velocidad nominal (n_2)	147.9 1/min
Torque (M_2)	191.0 Nm
Torque de parada	218.3 Nm
Valor característico de pérdida de potencia	0.953
Torque de aceleración máximo	385.0 Nm
Torque de freno de emergencia	513.0 Nm
Valor característico de corriente, parámetro SDS C03	1.76
Transformación	20.278
Velocidad máxima de salida (trabajo continuo)	3800 1/min
Velocidad máxima de salida (trabajo intermitente)	5000 1/min
Momento de inercia	7.98 kg cm^2
Backlash	10.0 arcmin
Rigidez	16.5 Nm/arcmin
Peso	30.2 kg
Eficiencia	0.97

Fuente: Stöber Antriebstechnik

ya que el motor y reducción fueron fabricados como unidad; así que no hace falta realizar ninguna transformación adicional.

$$J_t = W\left(\frac{p}{2\pi}\right)^2 + \frac{\pi L \rho R^4}{2} + J_m \tag{2.1}$$

Reemplazando valores en (2.1), se encuentra una inercia de 9.11 kg cm^2. El torque de aceleración es igual a la aceleración angular multiplicada por la inercia. Por cuestiones de seguridad y siguiendo las normas correspondientes, la velocidad de cierre de la mordaza no debe ser mayor a 5 m/s. Considerando un paso de 6 mm, la máxima velocidad angular será 87.26 rad/s. Estableciendo la velocidad angular máxima del sistema en 60 rad/s, y asumiendo que el motor debe llegar a dicha velocidad en 0.5 s; la aceleración para este periodo será aproximadamente 120 rad/s^2. Usando la ecuación (2.2), el torque de aceleración máximo resulta 0.11 N m. Esto indica que el torque de aceleración no supondrá ningún problema y que puede ser despreciado.

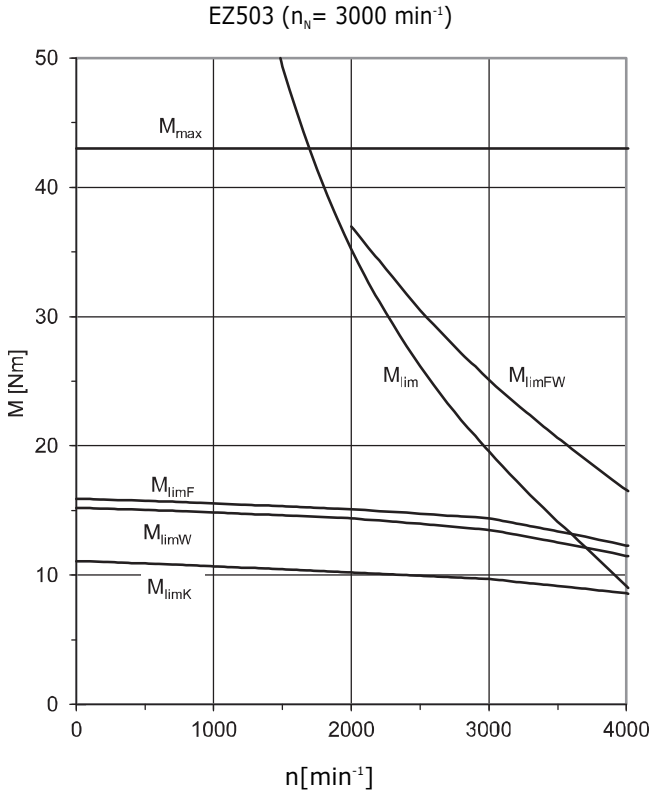


Figura 15. Curvas del motor
Fuente: Stöber Antriebstechnik

Donde :

M_{lim} =Límite de torque sin compensación de debilitamiento de campo
M_{limF} =Límite de torque con enfriamiento externo
M_{limFW} =Límite de torque con compensación de debilitamiento de campo
M_{limK} =Límite de torque con enfriamiento convencional
M_{limW} =Límite de torque con enfriamiento con agua
M_{lim} =Torque máximo

$$T_a = J_t \times \alpha \tag{2.2}$$

2.2.1. Base del motor

Para obtener la mayor eficiencia del motor y no reducir su tiempo de vida, los ejes deben estar alineados. Para poder medir el torque, un sensor debe ser añadido entre el motor y la carga. Además, se añadirá un eje capaz de aguantar el torque máximo. Esto significa que habrá varios componentes que alinear, y esto complica el diseño. La figura 16 muestra la estructura de instalación recomendada por el manual del sensor de torque. Obviamente esto es solo un esquema, los elementos reales deben ser diseñados.

El ensamblaje de la estructura del motor se muestra en la figura 17; en esta se ha enumerado los elementos más importantes.

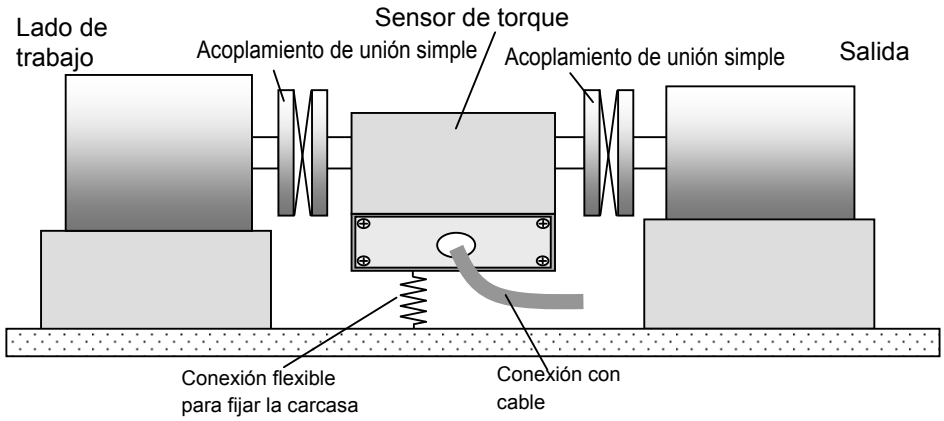


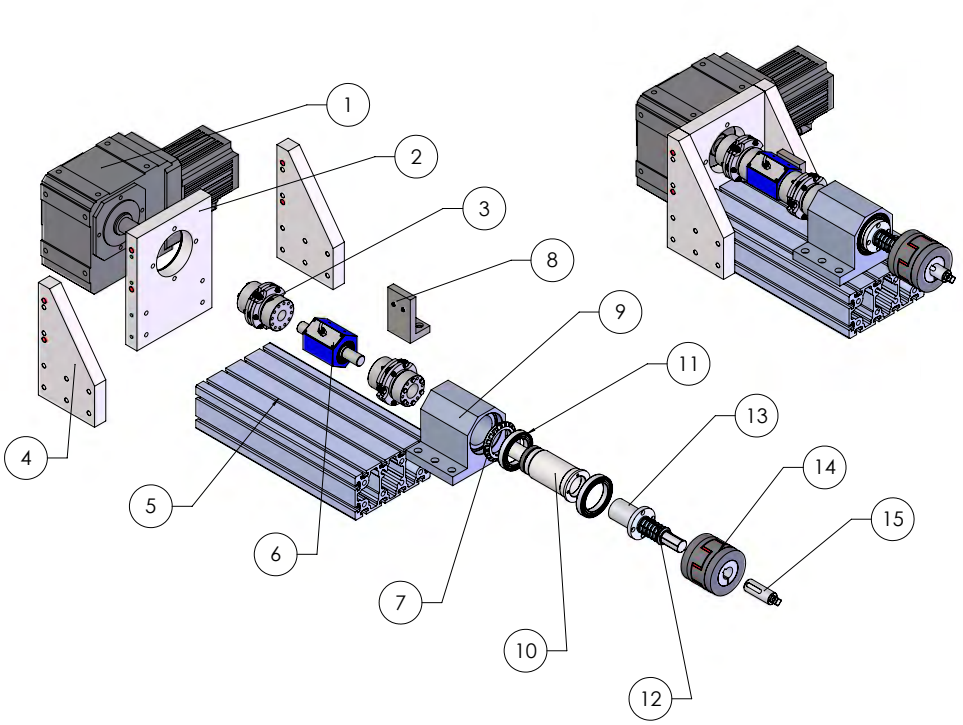
Figura 16. Diagrama de instalación del sensor de torque
Fuente: Manual de operación del sensor de momento de Lorenz Messtechnik

Para conseguir un ensamblaje fácil entre el motor y la mesa, se utilizó un perfil de aluminio con ranuras para roscas deslizantes como base. Las ranuras permiten asegurar piezas a la base y permite añadir nuevos elementos sin tener que modificarla. La parte número 2 fue diseñada de manera que tuviera un agujero de encaje para el alineamiento del eje del motor. Esto quiere decir que dicho agujero debe tener una cierta tolerancia en su medida, la cual es definida por el fabricante del motor como H7. La sección del motor diseñada para este ajuste tiene un largo de alrededor de 3 mm, pero la placa (elemento 2) tiene 30 mm de espesor. Aplicar una tolerancia H7 a todo el espesor no tendría sentido así que se estableció que solo una pequeña sección contaría con esta característica, tal y como se muestra en el plano de la pieza adjunto en los anexos. Para sujetar el motor con dicha parte se utilizó agujeros para pernos M8. Para mayor estabilidad, placas laterales fueron añadidas y estos tres elementos se conectaron con la base.

Como se indica en la vista esquemática, el sensor requiere una conexión flexible entre el sensor y la base. Además, acoples especiales deben ser utilizados, seleccionados de la misma empresa que fabrica los sensores de torque, Lorenz Messtechnik. Después de esta sección, el eje de torque debe ser conectado. Una base para el eje fue diseñada para este propósito; este es el elemento que se conectará al segundo acople del sensor. Una vez que el eje está asegurado con la base, el primero debe ser alineado, función desempeñada por la carcasa. En los planos de estos elementos se puede apreciar que, para todos los cambios de diámetro, se incluye un redondeo, chaflán o ranura. Esto tiene como finalidad reducir la concentración de esfuerzos que un cambio de diámetro implica.

Unos cuantos elementos más son mostrados en la figura 17. El objetivo del resorte es el de garantizar un buen acople entre el sistema del motor y las mordazas. El acoplamiento entre la estructura del motor y las mordazas fue seleccionado para que compense cualquier desalineamiento no detectado. Esta compensación se consigue gracias al elemento elástico que tiene en el centro, el cual se deforma cuando existe una deformación. Con esto se evita la presencia de momentos que puedan causar una deflexión en el eje. Después de este elemento se encuentra un pin conector con una cabeza cuadrada en la que entran los dados correspondientes a cada mordaza.

Para la base y carcasa del motor, la coaxialidad de las superficies de contacto es de suma importancia. Si estas superficies no son coaxiales, el eje no estará alineado. Por lo tanto, una tolerancia de 0.02 mm fue establecida para esta propiedad.



(1) Motor

(2) Base del motor

(3) Acoplamiento

(4) Base del motor, elementos laterales

(5) Base

(6) Sensor de torque

(7) Elementos de seguridad del eje

(8) Conexión flexible para el sensor

(9) Base del eje

(10) Carcasa del eje

(11) Rodamientos

(12) Resorte

(13) Eje de torque

(14) Acoplamiento cónico

(15) Pin conector

Figura 17. Motor con sus componentes
Fuente: Elaboración propia

2.2.2. Servocontrolador

En el primer capítulo se definió un servomotor como un motor con sensores integrados que permiten su control. Entonces, una vez que el motor ha sido definido, el controlador es seleccionado. Existen muchas posibilidades en esta área, y muchos factores que influyen en esta decisión. La mayoría de compañías que fabrican motores también producen módulos de control. Sin embargo, varias veces estos son, o muy complejos, o muy simples; o también pueden ser algo completamente diferente a lo que el encargado del proyecto está acostumbrado, lo que supone mucho tiempo de preparación por su parte.

En este caso, la decisión se basó en las recomendaciones de alguien que conoce del tema y ya ha trabajado con ALLMATIC anteriormente. Existe otro proyecto en la empresa que utiliza un motor eléctrico para operar una mordaza. El módulo de control utilizado en dicho proyecto proviene de la compañía B&R Automation. Según nuestro consultor, era posible emplear un controlador de la misma compañía, y, debida a que ya se tiene información sobre sus productos, la sugerencia fue aceptada.

2.3. Sistema

En este proyecto, el sistema es definido como el conjunto del banco de pruebas completo, y la mordaza a probar.

2.3.1. AMFE

Para asegurar el buen funcionamiento de un sistema es necesario analizar todos los posibles fallos. En los puntos anteriores ya se analizó las fallas estructurales y se explicó las decisiones en el diseño tomadas para evitarlas. En esta sección se trabajará con las fallas en la operación del sistema en general, para poder establecer parámetros de diseño que reduzcan su frecuencia o impacto. Para esto se realizó un AMFE o análisis modal de fallas y efectos. Después de llevar a cabo este proceso, se puede determinar si el sistema es seguro de implementar y, en caso no lo sea, qué medidas de seguridad o cambios en el diseño son necesarios para volverlo seguro.

El AMFE se muestra en la tabla 5. Las columnas "S", “O", "D” y “RPN"significan, respectivamente, nivel de severidad, nivel de ocurrencia (incidencia) , nivel de detección y número de prioridad del riesgo. Los tres primeros tienen un valor que va desde el 1 hasta el 10.

- Severidad

  Esta característica determina el efecto negativo que tiene el fallo en la función del sistema. Mientras peor sea el efecto, mayor será el valor de esta variable. Los valores de 9 y 10 están reservados para situaciones en las que se causa un daño al usuario.

- Incidencia

  Este parámetro mide la probabilidad de que el fallo ocurra, en una escala del 1 al 10, siendo 10 una posibilidad de 100 %.

- Detección

  Mide la facilidad con la que la falla puede ser detectada. El valor de 1 corresponde a una gran facilidad para detectar el error.

El número de prioridad del riesgo es el producto de estos tres valores e indica cuáles son los fallos a los que se le debe dar una mayor importancia. Según la tabla 5, los fallos a los que se le debe prestar una mayor atención son el de causar un daño a la mordaza y el de sobrecalentamiento del motor. Teniendo en cuenta que el máximo valor del RPN es 1000, un valor de 20 no es muy grave. De todas maneras se procurará minimizar la probabilidad de los errores que resultaron ser los más críticos. Sin embargo, no se utilizará métodos extraordinarios con este fin, pues su gravedad no lo amerita.

De las tres características, la severidad no puede ser modificada, por lo que el objetivo será a tratar de reducir la incidencia y/o mejorar la detección.

2.3.2. Algoritmo de control

ALLMATIC produce distintos tipos de mordazas. Como se mencionó en el primer capítulo, la característica fundamental es el multiplicador de fuerza; sin embargo, también se produce mordazas sin este elemento. Estas últimas tienen un funcionamiento muy simple: Se cierra la mordaza hasta que las bocas entran en contacto con la pieza

Tabla 5: Análisis modal de fallos y efectos

Falla	S	Efectos	Posible causa	O	Medidas de prevención actuales	Medidas de detección	D	RPN
Vibración	4	Vibración en todos los elementos	Patas no estables	2	Frenar las patas	Control visual, sonidos raros	1	8
Variación de la altura	5	Posición de las columnas elevadoras no está bloqueada	Error del programa	1	Depurar el programa	Controlar la corriente que llega a las columnas	1	5
Sobrecalentamiento o sobrecarga del motor	8	Carga muy alta	Bloqueo del motor, parámetros erróneos ingresados	2	El controlador tiene sistemas de seguridad integrados contra sobrecarga, el sistema no tiene que permitir el ingreso de valores no válidos	Sensores de fuerza y momento, control de corriente en el controlador del motor	1	16
Movimiento raro de las bocas	2	Problema del husillo, atascamiento de las bocas	Problema del husillo, atascamiento de las bocas	2	Revisar la mordaza y el husillo antes de las pruebas	Inspección	3	12
Daño de la mordaza	5	Fuerza o torque excesivos	Error de software, parámetros de entrada erróneos, mal funcionamiento de los sensores	2	Depurar el programa, no permitir entrada de valores erróneos, revisar los sensores	Control automático y manual	2	20
Peligro de accidente (atrapamiento, proyección de partículas)	9	Cubierta no está cerrada	Interruptor de seguridad no funciona	1	El sistema no debe empezar si la cubierta no está cerrada y bloqueada	Control automático y manual	1	9
Velocidad muy alta	5	Error del programa	Error del programa	1	Depurar el programa	Control automático y manual	2	10
Valores sin sentido	3	Husillo atorado, sensores no calibrados	Elementos extraños presentes en el husillo	2	El programa reconoce cuando los valores no son adecuados	Detectable por el programa	2	12
Apertura excesiva de la mordaza	7	El motor sigue funcionando después de alcanzar la apertura máxima	El sensor de inducción no funciona	2	Cuando el sensor se activa se debe detener el programa	Sensor	1	14

Fuente: Elaboración propia

y luego se aplica un mayor torque, normalmente a una velocidad menor, hasta llegar a la fuerza deseada. Cuando ya se realizó el trabajo requerido en la pieza, o es necesario cambiar de posición, se aplica un torque en el sentido opuesto para aflojar la pieza. El comportamiento del torque aplicado en relación con el ángulo de avance es uniforme, con algunas posibles irregularidades causadas por la forma del operador de realizar la fuerza. Este tipo de mordazas entonces, no presentará mayor problema en cuanto a su control.

En el caso más común de las mordazas con multiplicador de fuerza, la curva del torque en relación con el ángulo es más compleja. El multiplicador de fuerza no actúa durante todo el recorrido de las bocas sino solo después de que se ha alcanzado un torque determinado. Este torque es el necesario para realizar el acople del multiplicador con el sistema de husillo. En este punto donde se da dicho acople, el torque aumenta rápidamente, y, una vez que ya se ha realizado el acople, disminuye drásticamente. Este pico en el torque debe ser reconocido por el software para asegurar un óptimo control del sistema, ya que marca la separación entre dos etapas del proceso.

El proceso de sujeción de una pieza con estas mordazas se puede dividir en 4 etapas: cierre, compresión, afloje y apertura. El cierre comprende el recorrido de la boca móvil desde la posición abierta hasta el acople. La compresión es, como su nombre lo indica, el momento en el cual se aplica la fuerza de compresión sobre la pieza mediante el multiplicador de fuerza. El afloje consiste en retirar la fuerza que actúa sobre la pieza y el desacoplamiento del multiplicador marca su final. El recorrido de la boca móvil hasta alcanzar su posición inicial es lo que se conoce como apertura.

En cuanto a la boca móvil, se puede considerar que existen dos posiciones principales, las cuales serían la inicial y la final, mostradas en la figura 18 en ese orden. Durante la compresión, la boca se desplaza un diferencial de movimiento en la dirección X, y vice-versa en el afloje. Este movimiento no se ha graficado debido a que no es fácil de percibir.

Existe un caso aún más complejo, y es el de las mordazas tipo “DUO”. En estas se presiona dos piezas utilizando un solo multiplicador de fuerza. Existe una etapa de cierre y apertura por cada pieza; las otras dos etapas son comunes para las dos. Sin embargo, este caso puede ser considerado por el sistema de control como si fuera igual al anterior, así que no es necesario programar algo especial para esta situación.

Con estas consideraciones, se procederá a describir la estructura del sistema de control propuesta. Los diagramas de flujo correspondientes a las diversas etapas del sistema se encuentran en los anexos, así como la lista de variables utilizadas para representar las propiedades y/o estados en el sistema.

El sistema contará con dos modos de funcionamiento, el manual y el automático.

2.3.2.1. Modo manual

Este es el modo en el cual se entra al inicializar el sistema. Como su nombre lo sugiere, en este modo el usuario es capaz de controlar todos los parámetros de forma manual. En esta etapa, la cubierta estará desbloqueada, a fin de permitir al operador posicionar correctamente la mordaza y el sensor de fuerza. La altura de la mesa puede ser configurada según sea necesario.

Los parámetros básicos deben ser ingresados en esta etapa, tales como el paso

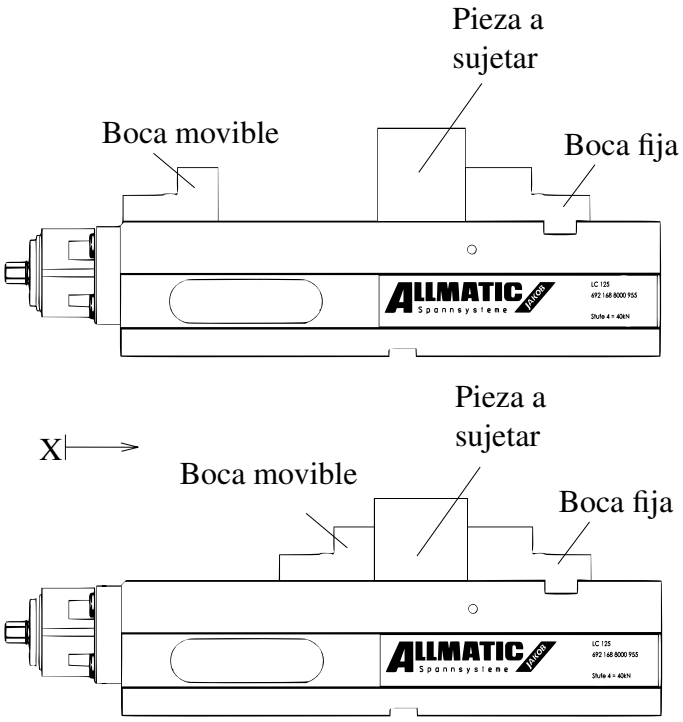


Figura 18. Posiciones de la boca
Fuente: Elaboración propia

del husillo, si tiene multiplicador de fuerza o no, el nombre de la mordaza, el responsable de la prueba, etc.

Dentro de este modo, el usuario debe ser capaz de accionar el motor para que la mordaza cierre o abra , al presionar una tecla. Estas teclas de avance (cierre) y retroceso (apertura) se diseñaron de manera que deban ser presionadas continuamente para que el motor se mueva. El fin de esta particularidad es el de aumentar la seguridad del sistema. Ya que la cubierta no está necesariamente cerrada, alguien podría acercar su mano a la zona de trabajo. El hecho de requerir una continua presión de las teclas, ocasiona que el usuario esté siempre pendiente de la máquina durante su operación, por lo que es más difícil que una posibilidad de peligro pase desapercibida.

Para prevenir una apertura excesiva de la mordaza, se instalará un sensor inductivo. El programa debe detener automáticamente el motor si es que este sensor es activado. Este es el único límite en el proceso de apertura en el modo manual. El límite en el otro sentido, es decir, en el cierre, está definido por un torque o una fuerza máximos, los cuales serán solicitados al inicio de este modo.

Durante el desarrollo del diseño, se acordó que, para el inicio del funcionamiento automático, la mordaza debe haber sido previamente tensionada. Esto quiere decir que el valor del torque o la fuerza máximos debe ser mayor que el que se produce en el pico que se da en el acople, explicado más arriba. De lo contrario, la mordaza no va a empezar en esta situación de pretensión. Estos valores se pueden encontrar realizando pruebas; es el usuario el que indicará que la mordaza ya ha sido tensionada y, por consiguiente, ya se puede pasar al siguiente modo. Las columnas elevadoras y la cubierta deben ser bloqueadas al final del modo manual.

2.3.2.2. Modo automático

En este modo, las 4 etapas –o 2 según sea el caso–, se dan una a continuación de otra. Teniendo en cuenta que en la posición inicial, la mordaza está tensionada, la primera etapa a realizar en este modo es el afloje. Después de cada etapa, el motor debe detenerse durante un pequeño momento antes de continuar. Este momento es igual para cada etapa excepto para la compresión, después de la cual se da una pausa cuya duración es definida por el usuario.

Como es de suponerse, los parámetros necesarios deben ser ingresados al inicio de este modo. De esta forma, y con los parámetros generales definidos en el programa, el sistema es capaz de reconocer el fin de cada etapa: El final del afloje se indica cuando la derivada del torque con respecto al ángulo recorrido supera cierto valor. El final del cierre es indicado también por esta derivada. La apertura termina cuando el motor ha girado la cantidad de vueltas indicada por el usuario, o el sensor inductivo se active. Esta última condición indica un fallo y el sistema debe pararse por completo. Por último, la compresión tiene más de una forma de ser controlada. Según lo que haya sido escogido, el motor debe proporcionar torque hasta que el torque de sujeción en la pieza, la fuerza de sujeción o el ángulo máximo de avance hayan sido alcanzados.

Ya que el único valor de posición que se puede obtener es el de la posición angular del motor, es necesario definir una posición como nuestro "cero" o referencia, de tal forma que haya algún valor con el que se pueda comparar el número de vueltas de apertura elegido. Esto también servirá para realizar cálculos posteriores.

En el caso de las mordazas sin multiplicador de fuerza, es decir, con solo 2 etapas; el proceso es simple. La etapa inicial es la apertura y luego sigue la compresión. Estas dos etapas no consideran en ningún momento el comportamiento causado por el amplificador de fuerza, por lo cual no hace falta hacer ninguna modificación de las etapas usadas en el modelo principal de mordaza.

En esta etapa, los valores de los sensores serán guardados cada cierta cantidad de ciclos, para poder trabajar luego con esos resultados. Independientemente de si son guardados o no, estos valores también deberán ser mostrados en todo momento.

Durante el modo automático, debe ser posible pausar el sistema. Cada vez que se realiza esta pausa, el motor debe ir hacia una *"Home Position"*. Dicha posición es igual a la posición a la que se llega al final de la apertura. Debido a esto, cada vez que se retoma el funcionamiento después de haber pausado el sistema, se debe iniciar ya no con el afloje, sino con el cierre; o en el caso de las mordazas sin multiplicador, con la compresión. La condición de pausa debe permitir al usuario modificar los parámetros que considere necesarios. Como es de suponer, no todos deben poder ser modificados, pues hay algunos que dependen de la geometría de la mordaza y, por lo tanto, no dependen del operador.

2.3.2.3. Detección y manejo de errores

En las secciones anteriores se mencionó el caso en el que el sensor inductivo se activa durante el modo automático. Existen otros casos considerados como errores, y el sistema debe ser capaz de detectarlos y detener el motor para evitar problemas adicionales. El análisis modal de fallos y efectos proporciona una lista general de

las situaciones consideradas como fallos. Como se puede apreciar en la columna correspondiente a “Medidas de detección” en la tabla 5, no todos los fallos pueden ser detectados por el sistema; sin embargo, esto no quiere decir que no se puedan prevenir o manejar. Existen otras formas de evitarlos, como, por ejemplo, informar al usuario de todos los riesgos presentes, incluir carteles con advertencias en el banco de pruebas, etc. Por suerte, son varios los errores que sí se pueden detectar por el sistema. Valores extraños o excesivos en los sensores indican la posibilidad de fallo. Con “valores extraños” se hace referencia a, por ejemplo, lecturas de fuerza de compresión cuando la mordaza está abierta; incremento en el valor del torque sin variación de la fuerza, etc.

Si bien es posible contar con una excelente capacidad de reacción ante fallas, lo ideal es que estas no se produzcan. Una de las razones por la que se dan los errores es el ingreso de variables incorrectas. Por esto, es necesario impedir que el usuario ingrese valores fuera de lugar. Con base en las normas internacionales, experiencia y sentido común; se puede elaborar una relación de valores máximos o rango de valores que son aceptables para cada parámetro. En el diseño del software está contemplado, para el proceso de ingreso de datos, que la etapa de entrada de valores no pueda ser abandonada si es que se ha ingresado valores equivocados o faltan valores necesarios para continuar.

Para agregar aún más elementos de seguridad, se estableció que las variables serían reiniciadas cada vez que el sistema se apague. Varias variables influyen en el control del motor; si el sistema es iniciado cuando las variables no están en su debido valor inicial, es posible que el motor se mueva cuando no es debido, causando daños al mecanismo o a la persona.

Capítulo 3
Construcción

En toda fabricación, las tolerancias y los ajustes son de gran importancia. Por eso, lo primero que se realizó después de recibir las piezas fue chequear todas las dimensiones importantes para asegurarse de que las tolerancias establecidas hayan sido respetadas. Dependiendo de la magnitud de la dimensión en particular, y la geometría a ser medida, se eligió los aparatos adecuados para el proceso de medición.

Durante la comprobación de medidas, la coaxialidad de la base del eje, cuya tolerancia había sido establecida en 0.02 mm, presentaba un error de 0.159 mm. Este es un valor aproximadamente 800 % mayor que el indicado. Se le informó al fabricante de la situación y este mandó una nueva pieza, cuyo nuevo valor de coaxialidad era 0.036 mm, mucho mejor que el anterior. Según Grote & Antonsson (2009), la tolerancia de coaxialidad para un eje depende de un parámetro llamado *"accuracy degree"*, o grado de precisión en español; con valores comunes de 5 a 10, mientras más pequeño es este valor, más preciso será el sistema. Para el diámetro con el que se trabajó, la tolerancia especificada corresponde a un grado 7. Si se cambia a 8, la tolerancia se vuelve 0.04, así que, según esto, el error obtenido en la nueva pieza es aceptable

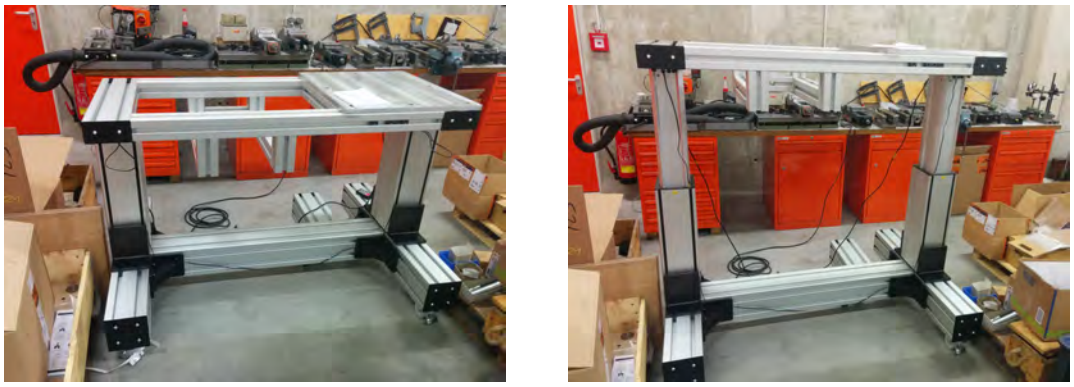
3.1. Mesa de trabajo

Para el montaje, hay algunas consideraciones que deben ser tomadas. Primero, como con cualquier estructura, hay que definir un orden. Si se pretende montar cualquier estructura, empezando por cualquier parte, es muy probable que se dé en algún momento la situación de que no se puede seguir adelante porque alguna de las piezas bloquea el ingreso de otra. El procedimiento de montaje utilizado en este caso fue el siguiente:

1. Montar las estructuras superior e inferior
2. Unir ambas estructuras con las columnas de altura variable, y controlar la altura. (Figura 19)
3. Montar placa de altura variable

4. Añadir el eje del motor

5. Fijar la caja de conexiones a la mesa



(a) Altura mínima (b) Altura máxima

Figura 19. Chequeo de las columnas elevadoras
Fuente: Elaboración propia

3.2. Base del motor

De la misma forma que se hizo en la mesa, se definió un orden para armar esta sección. Primero se montó el eje de torque, como se muestra en la figura 20a. Luego se montó la base general con las placas base del motor. Una vez hecho esto, se procedió a fijar el motor con el sensor de torque y se unió todo. La estructura resultante se muestra en la figura 20b.

(a) Eje de torque instalado en su base y carcasa (b) Todos los elementos montados

Figura 20. Estructura del motor
Fuente: Elaboración propia

Durante el montaje fue necesario tener siempre en cuenta las indicaciones en los manuales del fabricante. Por ejemplo, los pernos de los acoplamientos tenían que ser ajustados con un torque determinado para asegurar su buen funcionamiento. Un torque excesivo puede ocasionar roturas, y uno muy bajo puede ocasionar vibraciones.

3.2.1. Caja de conexiones

La caja de conexiones incluye los componentes de seguridad y control del banco de pruebas.

El diagrama de conexiones se muestra en el anexo. La estructura física de la caja de conexiones se puede apreciar en la figura 21.

(a) Interior de la caja de conexiones, parte superior

(b) Interior de la caja de conexiones, parte inferior

Figura 21. Caja de conexiones
Fuente: Elaboración propia

3.3. Transductores

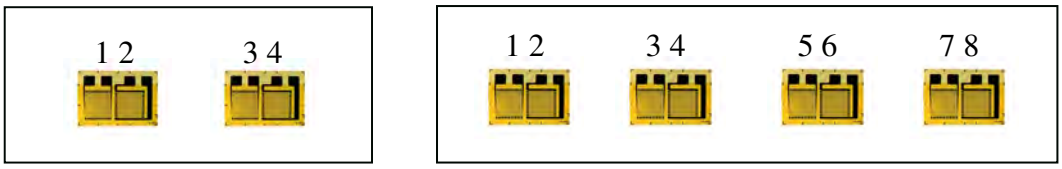
Para medir la fuerza obtenida con las mordazas, fueron construidos dos sensores (Figura 22). Estos fueron diseñados para soportar 40 y 150 kN respectivamente. Las galgas extensiométricas tienen una resistencia de 350 Ω. El material utilizado para los sensores tiene un módulo de elasticidad de 200 GPa.

(a) 40 kN

(b) 150 kN

Figura 22. Sensores de fuerza
Fuente: Elaboración propia

Ya que el área disponible en ambos sensores no es igual en los dos, se utilizó una disposición de galgas diferente para cada uno. Si los rectángulos mostrados en la figura 23, son el desarrollo del área lateral de los sensores, la disposición representada es la usada en los dispositivos.

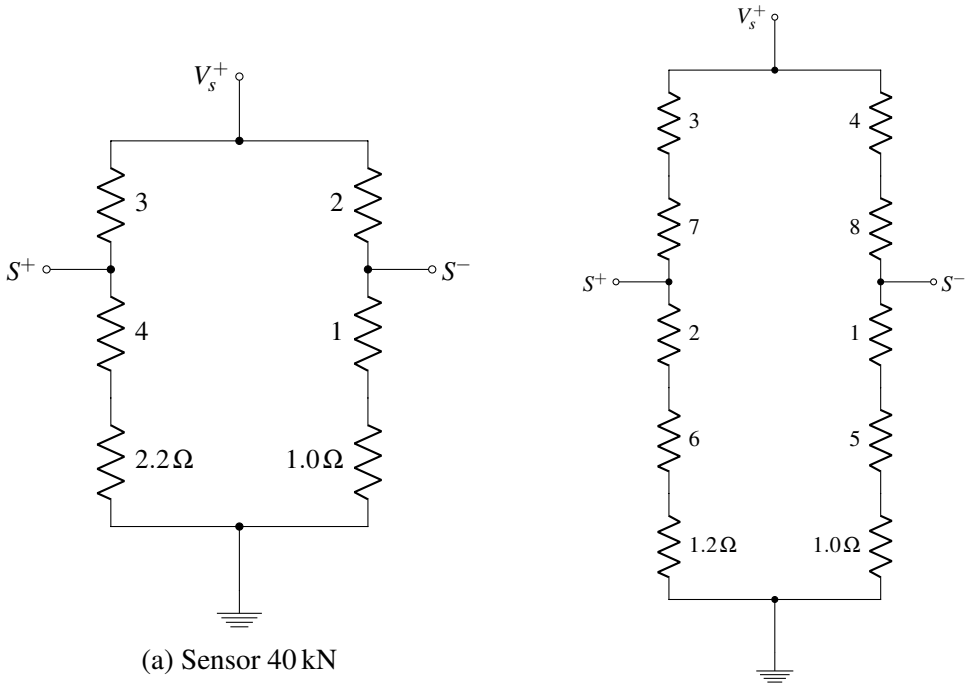


(a) Disposición de las galgas 40 kN (b) Disposición de las galgas 150 kN

Figura 23. Sensores de fuerza
Fuente: Elaboración propia

Los circuitos utilizados en los sensores se muestran en la figura 24. Las resistencias adicionales que se aprecian en dicha figura, fueron añadidas para calibrar los puentes. Conociendo la disposición de las galgas y el valor de su resistencia, es posible calcular el comportamiento de los sensores.

Para leer los valores de los puentes se utilizó un amplificador de señal. Encajado en una carcasa protectora, este amplificador fue acoplado a la estructura principal del banco de pruebas. La señal amplificada se conectó después a la caja de conexiones.



(a) Sensor 40 kN (b) Sensor 150 kN

Figura 24. Circuitos de las galgas extensiométricas
Fuente: Elaboración propia

3.4. Control

3.4.1. Componentes físicos

El controlador con los accesorios se encuentra en la caja de conexiones. El controlador está conectado con una computadora a través de Ethernet. La figura 25 muestra la configuración principal.

Los otros componentes se pueden encontrar en el diagrama de conexiones..

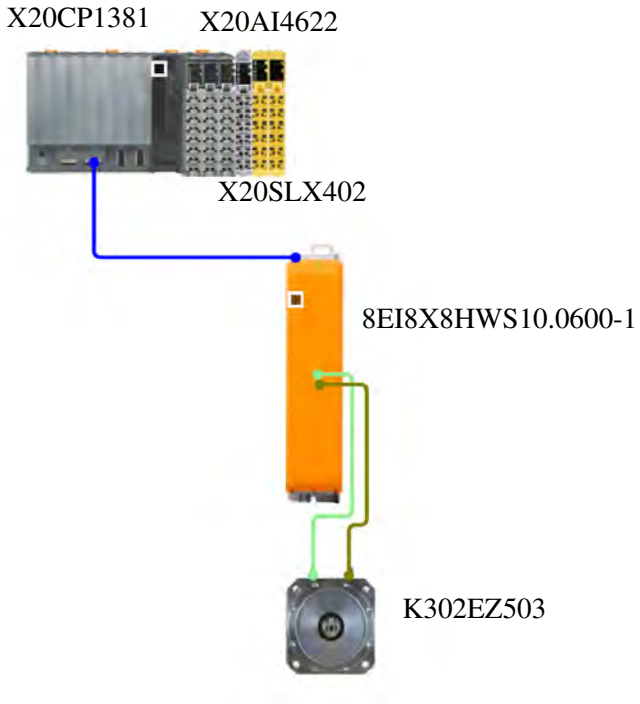


Figura 25. Componentes físicos principales
Fuente: Automation Studio 4.3

3.4.2. Configuración del controlador

Todos los componentes para el control del motor fueron comprados a la empresa B&R Automation. El motor es, sin embargo, de la empresa Stöber. Debido a esto, fue necesario brindarle al programa del controlador, los parámetros del motor.

El modo de control elegido fue el de posición. Esto quiere decir que el parámetro que utiliza el controlador para regular el movimiento del motor es la posición angular en que se encuentra, expresada en "Unidades" o *"Einheiten"* en alemán. Hay que tener en cuenta que el controlador solo regula el motor, sin tener en cuenta la reducción. Entonces, si se quiere controlar la salida real del sistema motor-reducción, tenemos que ingresar la relación de conversión en el programa. El software con el que el controlador es programado, requiere que se ingrese un número de las mencionadas "unidades", que equivalen a una vuelta. Para considerar la relación de transformación se definió que 3600000 unidades equivaldrían a 20278 vueltas. Por lo tanto, el eje de trabajo dará una vuelta cada 3600 unidades, lo que significa que cada unidad es igual a 0.1°.

El controlador tiene un tiempo de ciclo mínimo de 2 ms. El tiempo de ciclo para la clase de tarea del programa principal fue establecida en 4 ms para no sobrecargar al hardware. Se utilizó otros dos tipos de clase más, uno de 100 ms y otro de 50 ms. El primero se usó para el programa de la toma de datos, y el otro, para los procesos relacionados con la seguridad. Si bien la seguridad es de suma importancia, los relés de seguridad necesitan un tiempo para efectuar su función. Por esto, no tiene sentido requerirle al controlador llevar a cabo este tipo de procesos con una mayor velocidad, y, por consiguiente, utilizando más recursos; si los actuadores no van a beneficiarse con esto. Para tratar con asuntos más críticos de la seguridad, se utilizó un controlador dedicado adicional.

3.4.2.1. Controlador de seguridad

El controlador de seguridad es aquel marcado como X20SLX402 en la figura 25. Para programar este controlador es necesario utilizar un software adicional, de la misma compañía, llamado *'SafeDESIGNER'*. Además, este elemento solo puede ser programado utilizando *"Functional Blocks"*, o "bloques funcionales". El programa implementado se muestra en la figura 26.

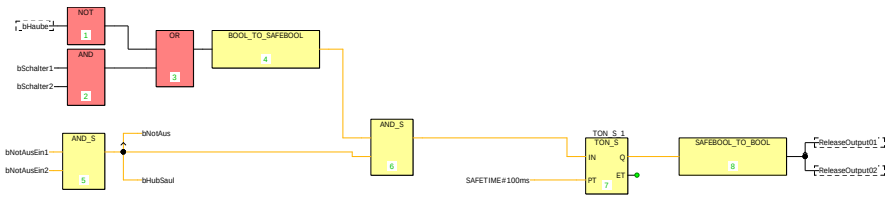


Figura 26. Programa del controlador de seguridad
Fuente: SafeDESIGNER

Este controlador tiene dos salidas digitales. Estas salidas controlan los relés de seguridad. El controlador se puede configurar de dos maneras, directa o vía *"SafeLogic"*. Esta última quiere decir que todo el programa de seguridad será exclusivamente controlado por el controlador de seguridad. Con la primera opción, el controlador general le manda una señal al de seguridad; luego, el programa en el controlador de seguridad decide si es seguro utilizar esa señal como salida o no. La opción elegida fue el modo directo, pues de esta forma se tiene una doble protección.

El programa de seguridad es sencillo. Hay dos variables que corresponden al estado del interruptor de seguridad de la cubierta (*bSchalter1* y *bSchalter2*), y otras dos que corresponden al interruptor de emergencia (*bNotAus1* y *bNotAus2*). Existen casos en los que la cubierta puede estar abierta sin causar peligro al usuario. El programa principal define esas situaciones y la variable que define esto es tomada por el programa de seguridad, esta variable es *bHaube* (figura 26). Si las condiciones son adecuadas, se manda una señal de salida positiva, lo cual activa los relés, dejando pasar la corriente y permitiendo el funcionamiento del sistema. En cualquier otro caso, la señal es negativa y, por ende, no se permite el funcionamiento del sistema.

3.4.2.2. Control remoto

Para que no sea necesario estar al costado del banco de pruebas para controlarlo, se implementó un servidor VNC. El entorno de visualización diseñado para el sistema puede, entonces, ser accedido desde todas las máquinas que se encuentran conectadas a la red de Allmatic.

Los datos obtenidos con las pruebas también se pueden descargar a distancia. El programa diseñado crea un documento de texto conteniendo los resultados de cada prueba. Este documento se guarda en la memoria interna del controlador. Para obtener este documento se configuró un servidor FTP, el cual también puede ser accedido por red.

3.4.3. Programación

El software para la programación y el control es también de la empresa B&R y se llama "Automation Studio". El lenguaje de programación para esto es conocido como

“texto estructurado”. Como ya se mencionó, el proyecto se dividió en 3 tipos de clase; en total, el proyecto cuenta con 6 programas. De estos, 2 son ejemplos que se pueden encontrar en las librerías de B&R, con pequeños cambios para adaptarlos a esta aplicación. El programa que trata las cuestiones de seguridad es bastante simple, pues el más importante fue programado en el controlador dedicado. El programa para la visualización y la toma de datos no tiene mucho que ver con el objetivo de este proyecto, por lo que no se explicará su funcionamiento. Todos los programas, a excepción de los ejemplos mencionados, se encuentran en los anexos.

La programación del programa principal del sistema fue diseñado como una “máquina de estado”. Esto significa que existe una variable que define el “estado” del sistema. Dependiendo de su valor, el sistema entra en una secuencia de comandos. Si las condiciones necesarias se cumplen, la variable que define el estado varía, lo cual lleva al sistema a otro estado.

El ejemplo sacado de las librerías de B&R, para el control de una estructura con un solo eje, tiene una variable de control (*“Control-Structure”*). Dicha variable se llama “basic_Typ” y está definida como se explica en la tabla 6.

Tabla 6: basic_typ

Nombre	Variable	Tipo	Descripción
basic_axisState_typ	Disabled	BOOL	si es TRUE, el eje está en estado “Disabled”
	StandStill	BOOL	si es TRUE, el eje está en estado “StandStill”
	Homing	BOOL	si es TRUE, el eje está en estado “Homing”
	Stopping	BOOL	si es TRUE, el eje está en estado “Stopping”
	DiscreteMotion	BOOL	si es TRUE, el eje está en estado “Discrete Motion”
	ContinuousMotion	BOOL	si es TRUE, el eje está en estado “ContinuousMotion”
	SynchronizedMotion	BOOL	si es TRUE, el eje está en estado “SynchronizedMotion”
	MoveAdditiveDone	BOOL	movimiento aditivo terminado
	MoveAbsoluteDone	BOOL	movimiento absoluto terminado
	HomeDone	BOOL	homing terminado
	ErrorStop	BOOL	si es TRUE, el eje está en estado “ErrorStop”
basic_command_typ	Power	BOOL	encender el controlador
	Home	BOOL	referenciar el eje
	MoveAbsolute	BOOL	moverse hacia una posición definida
	MoveAdditive	BOOL	desplazarse una distancia definida
	MoveVelocity	BOOL	empezar un movimiento con una velocidad definida
	Halt	BOOL	detener todo movimiento activo
	Stop	BOOL	detener todo movimiento activo, siempre y cuando su valor sea TRUE
	MoveJogPos	BOOL	moverse en dirección positiva mientras su valor sea TRUE
	MoveJogNeg	BOOL	desplazarse en posición negativa mientras su valor sea TRUE

Tabla 6 – (continuación)

Nombre	Variable	Tipo	Descripción
	ErrorAcknowledge	BOOL	borrar errores activos
basic_parameter_typ	Position	REAL	posición objetivo para el comando MoveAbsolute
	Distance	REAL	distancia para el comando MoveAdditive
	Velocity	REAL	velocidad para el comando MoveVelocity
	Direction	USINT	dirección para los movimientos
	Acceleration	REAL	aceleración para los movimientos
	Deceleration	REAL	desaceleración para los movimientos
	HomePosition	REAL	posición objetivo para referenciar el eje
	HomeMode	USINT	modo para el homing
	JogVelocity	REAL	velocidad para el movimiento de jogging
basic_status_typ	ErrorID	UINT	número de identificación de cualquier error ocurrido
	ErrorText	STRING[79][0..3]	texto de error
	ActPosition	REAL	posición actual del motor
	ActVelocity	REAL	velocidad actual del motor
	DriveStatus	MC_DRIVESTATUS_TYP	estado actual del eje
basic_typ	Command	basic_command_typ	comando
	Parameter	basic_parameter_typ	parámetros
	Status	basic_status_typ	status structure
	AxisState	basic_axisState_typ	estado del eje

Fuente: Automation Studio 4.3

Como se puede notar en esta tabla, existen 2 salidas y 2 entradas. La estructura de comando del ejemplo es tal, que cuando un comando es TRUE, el proceso correspondiente es llevado a cabo, sin embargo, el comando no vuelve a su valor inicial (FALSE) después de que el proceso ha terminado. Esto hace que sea necesario reiniciar el valor de estos comandos en el programa principal. La estructura "axisState" permite ver cuando el proceso ya terminó, con esto se puede reiniciar los comandos como es requerido.

Las señales analógicas siempre tienen ruido. Este ruido dificulta el control, así que el programa inicia con un filtro. El filtro utilizado es el de la media móvil. Esto significa que el valor obtenido en cada ciclo es promediado con los valores anteriores. La media móvil es un filtro pasabajo, por lo cual el ruido, el cual tiene altas frecuencias, es eliminado.

Después de algunas pruebas, fue posible encontrar una gráfica característica de las mordazas. Con estas gráficas se llegó a la conclusión de que iba a ser necesario trabajar con alguna derivada para poder controlar el proceso. La derivada escogida fue la del momento. Entonces, el siguiente paso en el programa principal es calcular la derivada.

Después de hallar todos los parámetros necesarios, se verifica si existe algún error, en cuyo caso, todos los comandos se establecen como FALSE. Los primeros estados del sistema son de inicialización. En estos, el controlador es encendido; y el motor, referenciado. El modo de referenciación (*homing*) es *'Direct'*, lo que quiere decir que la posición actual se convierte en el cero del sistema.

3.4.3.1. Modo manual

El sistema entra después en el modo manual. En este modo, el motor se mueve con un movimiento 'jog'. Esto significa que el motor se moverá si y solo si el comando es TRUE. Para evitar accidentes, la velocidad de giro fue establecida en $8.\overline{3}$ RPM. Adicional a esto, se implementó una medida como se explica en 2.3.2.1.

Hay 3 modos de control para el funcionamiento del sistema: fuerza, momento y ángulo –posición–. Durante el modo manual es necesario guardar 2 posiciones angulares, estas son la posición inicial y la posición final. Dichas posiciones pueden ser guardadas al presionar una tecla, aunque también existe la posibilidad de guardarlas automáticamente.

En el modo de control según fuerza o momento, cuando el valor de estas variables es mayor que el escogido como el valor objetivo, la posición en que se da esto es guardada como la posición final para el sistema. Esta condición de que el valor sea mayor que el objetivo tiene que ser cumplida también cuando se graba la posición manualmente, de lo contrario habrá errores en el modo automático. La razón de esto se explicará más adelante.

La posición inicial se graba automáticamente cuando se da el desacople del amplificador de fuerza. A esta posición se le resta la apertura –parámetro ingresado por el usuario, en milímetros–, y el resultado se toma como posición inicial. Si se quiere guardar manualmente esta posición, se tiene que decidir cuál es la posición adecuada. En este caso, la apertura ingresada por el usuario no se toma en cuenta. Para las mordazas sin amplificador de fuerza solo se puede grabar manualmente esta posición.

Como se mencionó, esto solo vale para los modos de fuerza o momento. Para el control por ángulo existe otro algoritmo. Este tipo de control está pensado para otro modelo de mordaza, en el cual se tiene un tope de momento. Esto significa que, una vez alcanzado cierto momento, se activa un freno. En este caso, sería necesario un momento muchísimo mayor para seguir dando vueltas. Esto es reconocido por el sistema y la posición en la que el momento se dispara es guardada. Luego, en el modo automático, se le resta el valor ingresado como 'ángulo', y esta nueva posición se usa como posición final. Si bien este modo de control está pensado para las mordazas con tope de momento, también puede utilizarse con otros tipos de mordazas; sin embargo, para ello es necesario guardar la posición final manualmente. La posición inicial se guarda igual que en los otros modos de control.

3.4.3.2. Modo automático

Después de que los parámetros necesarios han sido guardados o calculados, y estos son válidos, se puede pasar al modo automático. El modo automático inicia con el cierre de la mordaza. Una vez que el valor deseado de fuerza, momento o posición es alcanzado, se detiene el motor. El tiempo que el motor se detiene en esta posición es ingresado por el usuario. Después de pasado este tiempo, se abre la mordaza hasta la posición inicial, en la cual se espera por un tiempo también ingresado por el usuario.

Es necesario aclarar algunos puntos sobre el párrafo anterior. En el modo manual, el motor se mueve con un movimiento "absoluto". En el modo de control según ángulo, el comportamiento es sencillo. El motor se mueve hacia la posición final y luego hacia la inicial. El caso del control según fuerza o momento es más complicado. En estos modos de control, el motor **no** debe llegar hasta la posición final. La posición final sirve como límite de seguridad, es por esto que, en el modo manual, esta posición es guardada cuando el valor de fuerza o momento es mayor que el valor objetivo o deseado. Cuando el sistema reconoce que el valor de la variable de control es igual al objetivo, detiene el motor; sin embargo, le toma un tiempo al sistema físico para llevar a cabo esta operación, por lo que existe un error. Entonces, la posición final debe contemplar este posible error. El sistema mide el error en cada ciclo y corrige el set point para alcanzar luego el valor deseado.

En el caso de que, debido al error mencionado, o a algún cambio en el comportamiento de la mordaza, se llegue a alcanzar la posición final, el sistema se detendrá y mostrará un error. Para este caso, se implementó la posibilidad de corregir la posición final y seguir trabajando, de lo contrario sería necesario volver al modo manual para guardar otra posición como posición final.

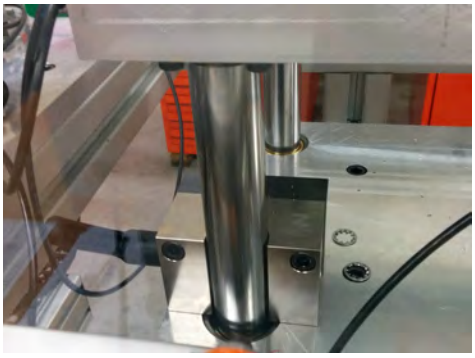
En el modo automático se graba ciertos valores por cada ciclo. Estos son, la fuerza y el momento máximos, la posición final, el acople y desacople del amplificador de fuerza, y la posición inicial.

3.5. Manual de operación

3.5.1. Montaje de la mordaza

Para montar la mordaza en el banco de pruebas se tiene que abrir la cubierta protectora. La altura de la placa variable se debe ajustar, utilizando el mecanismo de husillo, moviendo la palanca que se muestra en la figura 27a. Para fijar la altura, se debe utilizar los frenos, como se muestra en la figura 27b.

(a) Palanca



(b) Freno

Figura 27. Montaje
Fuente: Elaboración propia

Si la mordaza lo permite, las tuercas de centrado deben ser ingresadas en la ranura correspondiente.

Si la mordaza cuenta con un amplificador de fuerza, esta debe ser empujada hasta que el resorte del eje se encoja **mínimo** 15 mm. La figura 28 muestra la posición final que debe tener la mordaza.

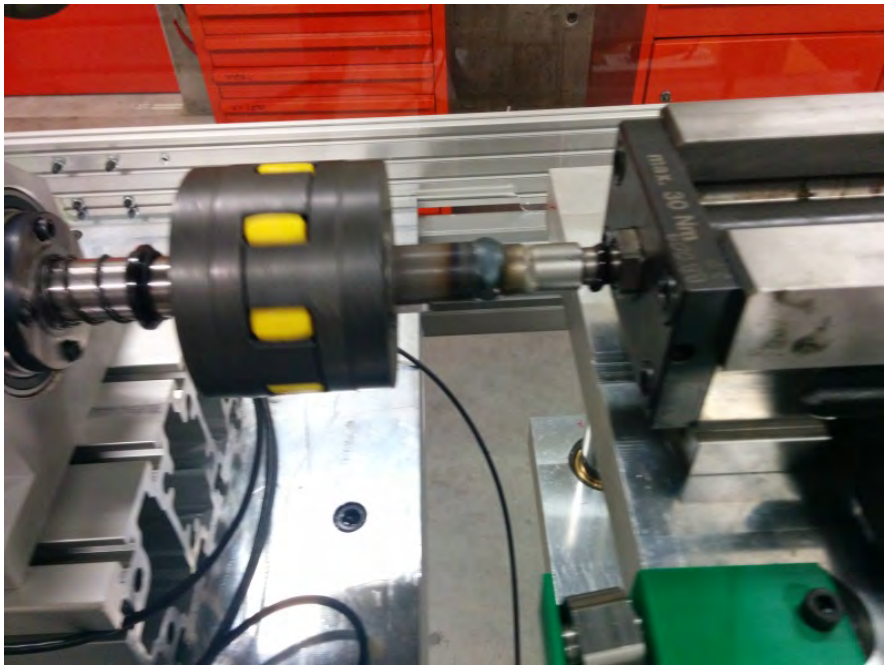


Figura 28. Mordaza con amplificador, empujando 15 mm
Fuente: Elaboración propia

Por último, la mordaza debe fijarse con ayuda de los pernos ajustadores.

3.5.2. Encendido

Después del encendido del interruptor principal del banco de pruebas, se debe esperar hasta que la lámpara se encienda de color verde (figura 29)

Figura 29. Lámpara
Fuente: Elaboración propia

Nota: Si el interruptor de emergencia está apretado, la lámpara se quedará de color rojo y no será posible iniciar ningún movimiento.

ADVERTENCIA: ¡La alimentación de 3 x 400 V **y el tomacorriente de** 230 V **tienen corriente incluso cuando el interruptor principal está apagado!**

3.5.3. Inicio

La HMI–*Human Machine Interface: Interfaz humano-máquina*– para el control del sistema es accesible a travez de un servidor VNC, como se muestra en la figura 30. La dirección del servidor es 192.168.27.101, y el puerto es el predeterminado (5900).

En la primera página de la HMI (figura 31) se debe ingresar los parámetros necesarios para la prueba. La explicación de los parámetros se da en la tabla 7. Se puede escoger si se desea utilizar una lista de parámetros predefinidos, o si se prefiere ingresar todos los parámetros manualmente. Dicha lista contiene los límites máximos de fuerza y momento, el paso de la rosca, y si la mordaza tiene o no un amplificador de fuerza. El sistema empieza con la opción de elegir desde la lista.

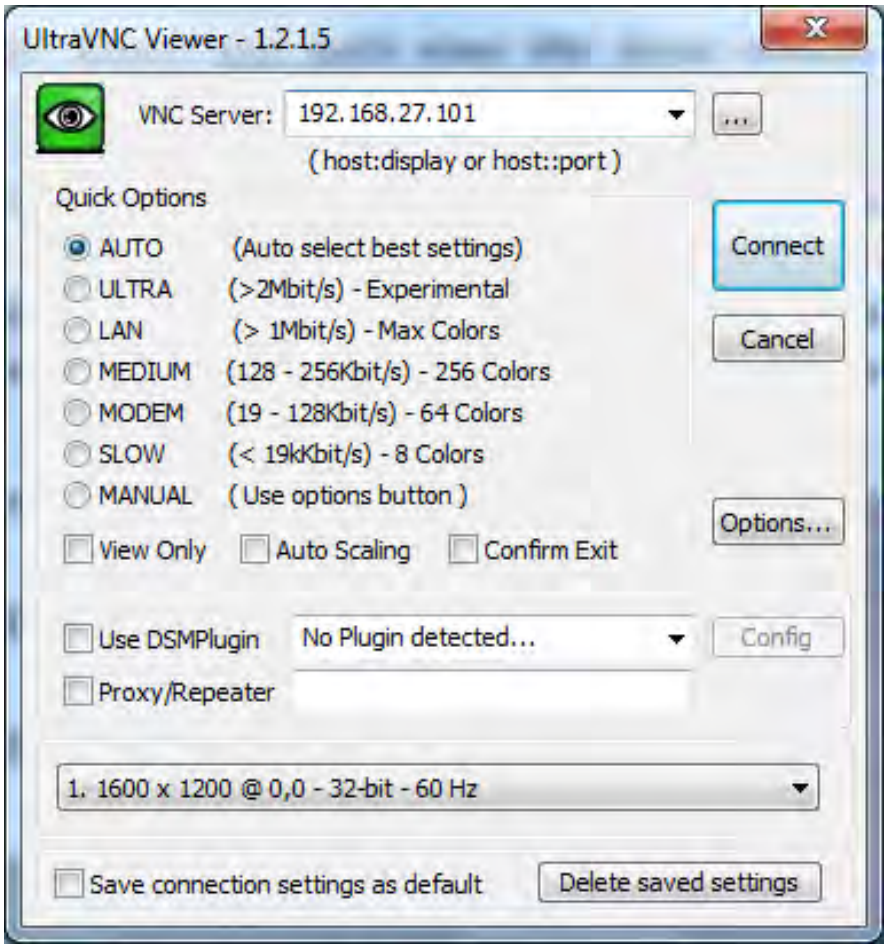


Figura 30. Acceso al servidor VNC
Fuente: Elaboración propia

Tabla 7: Parámetros a ingresar

Parámetro	Significado
Kraft Grenze	Valor máximo de fuerza permitido
Drehmoment Grenze	Valor máximo de momento permitido
Konstrukteur	Nombre del diseñador/usuario
Versuch	Descripción cualquiera de la prueba
Max Kraft	Valor deseado en el control según fuerza
Max Drehmoment	Valor deseado en el control según momento
Winkel	Valor deseado en el control según ángulo
Steigung	Paso de la rosca de la mordaza
Öffnungsweite	Determina cuánto se abre la mordaza
Hübe	Número de repeticiones
Spann Dauer	Duración de la posición cerrada
Ungespannte Pos. Dauer	Duración de la posición abierta
Kraftmessdose	Cuál sensor se utilizará
Steuerungsart	Modo de control

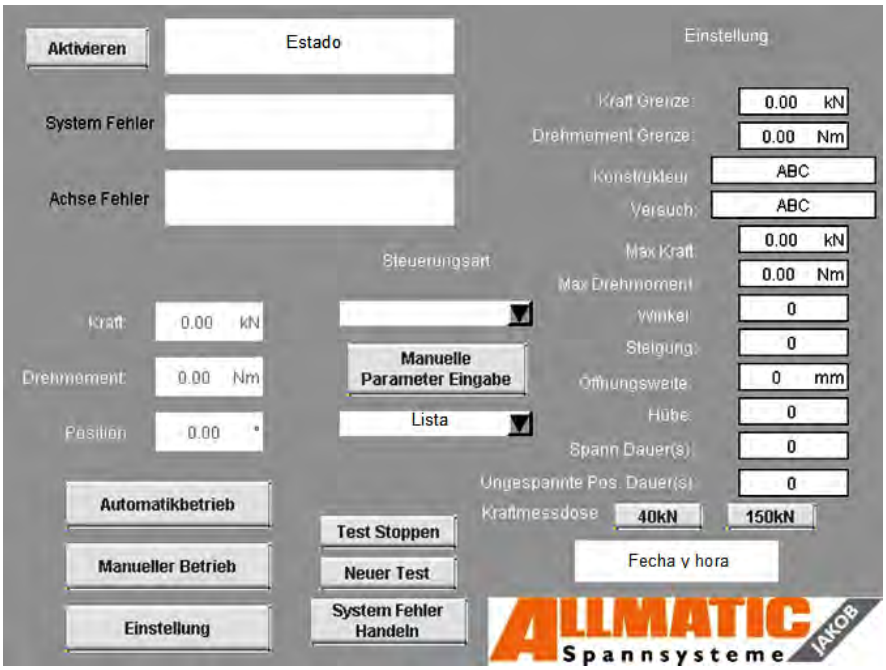


Figura 31. HMI:Página de inicio
Fuente: Elaboración propia

En la entrada de datos manual, y con los otros parámetros que no están definidos en la lista; si no se ingresa valor alguno, existen unos valores predefinidos. Estos valores están ahí para que el sistema no entre inmediatamente en un estado de error, pero eso no quiere decir que deban o puedan ser usados. El usuario debe verificar que son adecuados para la prueba en cuestión.

Entre los parámetros *"Max Kraft"*, *"Max Drehmoment"* y *"Winkel"* –fuerza máxima, momento máximo y ángulo–, solo es necesario ingresar uno, dependiendo del modo de control elegido.

Después de que los parámetros hayan sido ingresados o seleccionados, se debe pasar a la página del modo manual –*Manueller Betrieb*–.

3.5.4. Modo manual

3.5.4.1. Encendido del controlador

A pesar de que el banco de pruebas ya está encendido, el controlador todavía no lo está. Una vez en la página "Modo manual" (figura 32) se debe presionar las teclas *"Aktivieren"* y *"Antrieb Ein"*, para encenderlo. Antes de eso, el estado debe mostrar *"WARTEN"*. Después de encender el controlador, el estado debe mostrar *"HOMING"*, lo cual indica que se debe referenciar el motor. Para esto, se debe presionar la tecla *"Referenzieren"*. Una vez hecho esto, el estado debe cambiar a *"Manueller Betrieb"*. Así mismo, la tecla *"Aktivieren"* se desactivará.

El modo manual permite al usuario mover el motor libremente, siempre y cuando no se excedan los valores máximos. Este modo está pensado para encontrar y guardar las posiciones inicial y final; así como para abrir y cerrar la mordaza según lo requiera el usuario.

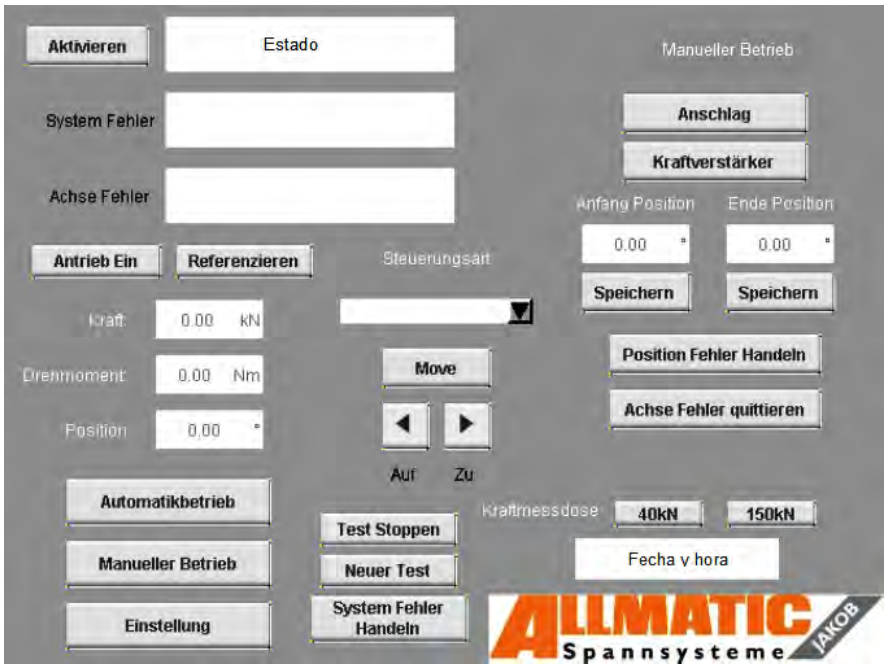


Figura 32. HMI: Modo manual
Fuente: Elaboración propia

3.5.4.2. Movimiento

Para empezar un movimiento, se debe presionar otra vez la tecla *"Aktivieren"*. La dirección puede ser escogida según el deseo del usuario, haciendo click en las teclas etiquetadas como *"Auf"*–abrir– y *"Zu"*–cerrar–. El motor solo se moverá si es que la tecla *"Move"* está siendo presionada. Si se deja de presionar la tecla, el movimiento se detendrá.

3.5.4.3. Guardar la posición final e inicial

Es necesario guardar estas posiciones para empezar el modo automático. Esto se puede realizar manual o automáticamente.

3.5.4.3.1. Manual

Para guardar una posición como inicial o final, simplemente hay que presionar la tecla *"Speichern"* cuando la posición actual sea la deseada. Hay dos de estas teclas, para guardar la posición inicial se debe presionar aquella que está debajo de *"Anfang Position"*–posición inicial–. Por consiguiente, la otra tecla corresponde a la posición final –*"Ende Position"*–. Si no hubo ningún problema, debería mostrarse las posiciones guardadas.

3.5.4.3.2. Automático

- La **Posición inicial** solo puede ser guardada automáticamente en las mordazas que poseen un amplificador de fuerza. Para que se guarde, se debe cerrar la mordaza hasta **generar un poco de fuerza**, 5 kN deberían ser suficientes. Luego se debe abrir la mordaza hasta que el amplificador de fuerza se desacople. Este punto es reconocido por

el sistema. Con el dato ingresado como *"Öffnungsweite"*–apertura–, se calcula la posición inicial. *Esta posición es referencial, dependiendo del amplificador, la apertura puede o no ser igual a la escogida.*

- La **posición final** se guardará automáticamente cuando la fuerza o el momento máximos más un 10 % –tolerancia de error– se haya alcanzado; en el modo de control de fuerza o momento respectivamente. En el modo según ángulo, la posición se guarda cuando se reconoce un tope en momento, el cual ya se explicó más arriba.

Una vez que se tienen las posiciones necesarias, se debe pasar al modo automático.

3.5.5. Modo automático

3.5.5.1. Pasos previos

Ahora llegamos a la página mostrada en la figura 33. Para poder empezar este modo, se debe confirmar que las posiciones guardadas son correctas, presionando la tecla *"Position Bestätigen"*–Confirmar posición–. Si no hay errores, el estado debe cambiar a *"KRAFT"*–fuerza–, *"UMDREHUNGEN"*–vueltas/ángulo– o *"DREHMOMENT"*–momento–, dependiendo del modo de control escogido.

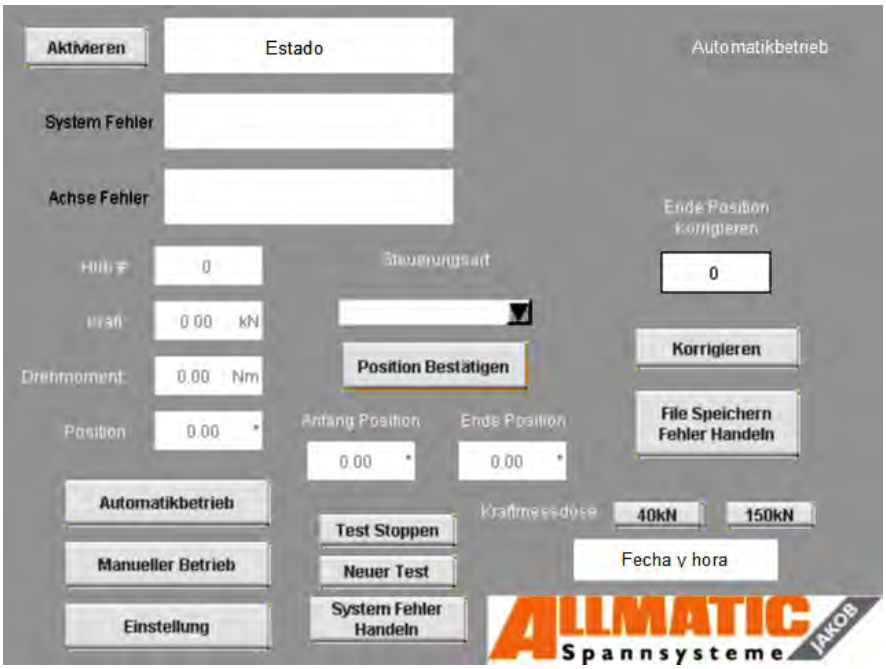


Figura 33. HMI: Modo automático
Fuente: Elaboración propia

3.5.5.2. Empezar, pausar o cancelar una prueba

Cuando se quiere iniciar una prueba, se debe volver a presionar la tecla *"Aktivieren"*. Si se desactiva esta tecla, se pausará la prueba. Durante la prueba, esta puede ser cancelada presionando la tecla *"Test Stoppen"*. En este último caso, el sistema pasa automáticamente al modo manual.

3.5.5.3. Prueba terminada

Cuando una prueba o test ha terminado, el estado debe ser "modo de control *FERTIG*". Aquí se puede empezar una nueva prueba presionando *"Neuer Test"*. El modo manual también puede ser accedido desde este estado, desactivando la tecla *"Position Bestätigen"*.

3.5.6. Manejo de errores

Existen algunos casos de error en los que no se necesita ningún manejo extraordinario, por ejemplo, cuando no se ha guardado las posiciones. Estos errores no llevan al sistema a un estado de error. Para aquellos casos en los que sí se va a un estado de error, se diseñó 5 procesos para manejar estos errores.

3.5.6.1. Error de sistema–*System Fehler*

Después de su detección, el sistema necesita reiniciarse. Esto significa que el sistema regresa al estado inicial: *"WARTEN"*. El controlador se apagará, y se necesita un nuevo referenciamiento. El usuario tiene permitido abrir la cubierta protectora para corregir el error.

Si es necesario mover el motor para corregir el error, se debe presionar la tecla *"System Fehler Handeln"*. Esto permite al operador ir hacia el modo manual.

PRECAUCIÓN: ¡Al presionar esta tecla, ya no se toma en cuenta los valores máximos de seguridad! ¡Existe riesgo de dañar en el motor y/o la mordaza, y de daño al usuario!

3.5.6.2. Error en el eje–*Achse Fehler*

Estos errores no tienen nada que ver con el sistema del banco de pruebas. Estos problemas pasan, por ejemplo, cuando se presiona el interruptor de emergencia y, por lo tanto, se corta la fuente de corriente al controlador.

Al presionar la tecla *"Achse Fehler quittieren"*, el error mostrado desaparecerá, obviamente, siempre y cuando la causa del error ya se haya corregido. Esto debe repetirse hasta que ya no se muestre ningún error en la HMI.

3.5.6.3. Guardado de datos–*File Speichern*

Cuando se da algún error en el guardado de datos, se detendrá el sistema. El error más común se da cuando el nombre del archivo que se intenta crear ya ha sido utilizado. El nombre tiene el siguiente formato: Fecha_constructor-_prueba_nombredelamordaza.txt.

Para corregir este tipo de error, se debe cambiar el nombre del constructor o de la prueba, o eliminar el archivo de texto existente. Se puede acceder a los archivos mediante un servidor FTP. **La dirección de este servidor es 192.168.27.101. El nombre de usuario es "admin", y la contraseña, "1234"**. Como se muestra en 34, el directorio donde se encuentran los datos es "*F:*\".

Para ver los archivos, se debe hacer click derecho en el archivo deseado y seleccionar *"Ansehen/Bearbeiten"*–ver/modificar–, con el cliente FTP "FileZilla", como se muestra en la figura 34. Si se selecciona otra opción durante el funcionamiento del sistema, se puede malograr la toma de datos. **¡Este caso no es reportado!**

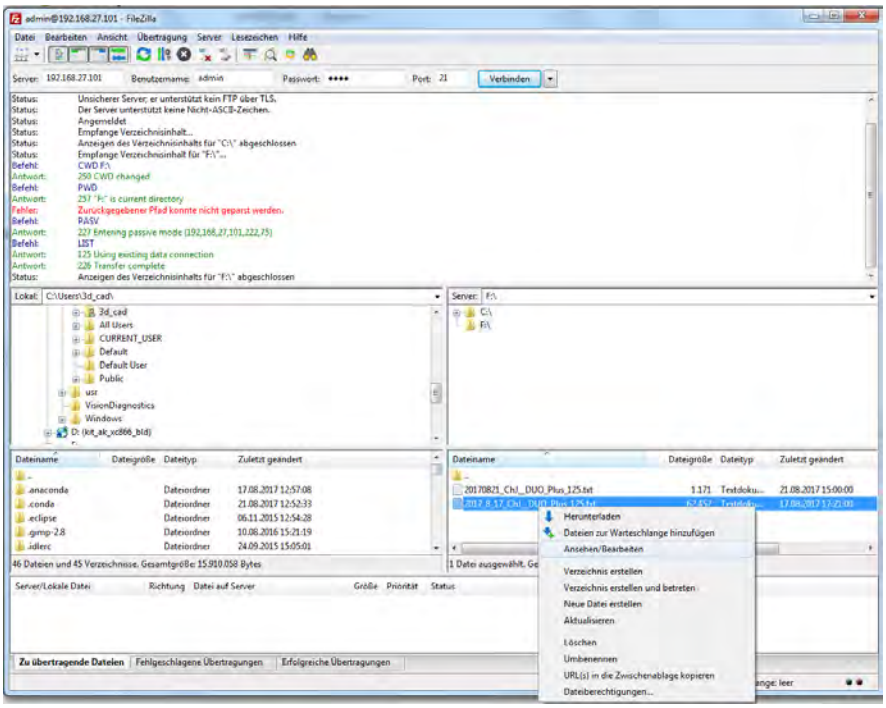


Figura 34. Servidor FTP
Fuente: Elaboración propia

Para eliminar el archivo, se debe seleccionar *"Löschen"*–eliminar–. Una vez que se corrigió la causa del error, se debe presionar *"File Speichern Fehler Handeln"*. El sistema debe ir ahora hacia un estado libre de error.

Es posible que el error no se haya corregido incluso después de hacer esto. Si eso pasa, se debe reiniciar por completo el banco de pruebas.

3.5.6.4. Posición final–*Ende Position*

Este proceso lidia con el error en el modo de fuerza y momento, cuando se alcanza la posición final. Puede ser que esto pase cuando no existe ningún error, o cuando el comportamiento de la mordaza haya variado debido al desgaste.

Con la tecla *"Korrigieren"* se puede añadir un ángulo determinado a la posición final, para evitar que se detecte este error. Después de hacer esto se puede retomar la prueba.

3.5.6.5. Error de posición–*Position Fehler*

Existe un sensor inductivo para evitar que las bocas se muevan excesivamente, ya sea al abrir o cerrar. Cuando el sensor no detecta la boca, el sistema se detiene

Para corregir este error, sin apagar el sistema completo, se puede presionar la tecla *"Position Fehler Handeln"*. Así se puede ir al modo manual y mover libremente el motor, para llegar hasta una posición válida.

Cuando el sensor inductivo vuelve a reconocer la boca, este error desaparece.

PRECAUCIÓN: ¡Después de presionar esta tecla, ya no se toma en cuenta los límites de posición! ¡Existe peligro de dañar la mordaza o el motor, y de daño al usuario!

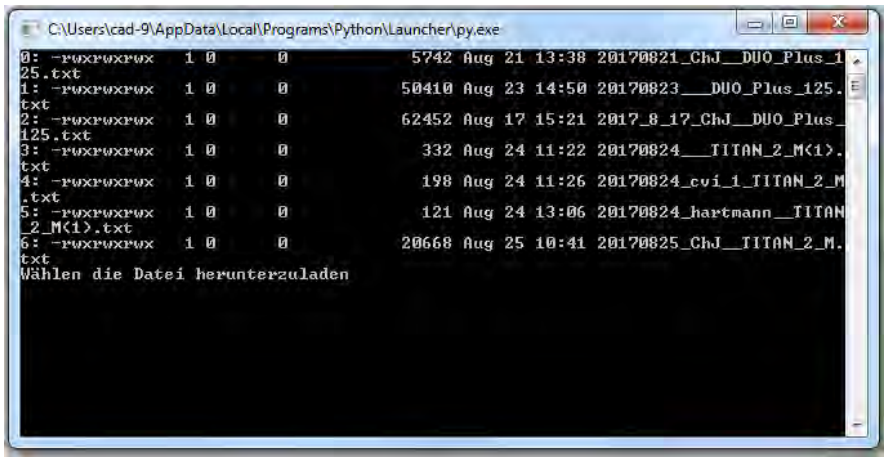


Figura 35. Programa para leer y graficar los datos
Fuente: Elaboración propia

3.5.7. Ver los datos

Para ver los datos, existe un programa adicional llamado “DateiLesenUndZeichnen.py”, mostrado en la figura 35. Este programa muestra los archivos de texto que se encuentran en el servidor FTP. Se puede elegir el archivo deseado ingresando su número correspondiente, lo cual devuelve un archivo en Excel mostrando los datos contenidos en el archivo, y un PDF mostrando unas gráficas de estos datos.

Capítulo 4
Resultados

4.1. Comparación con los anteriores bancos de pruebas

Los bancos de pruebas anteriores eran bastante simples e ineficientes, como ya se mencionó anteriormente. Aparte de eso, no existe documentación alguna sobre estos módulos. La figura 36 muestra un dibujo de uno de los módulos anteriores.

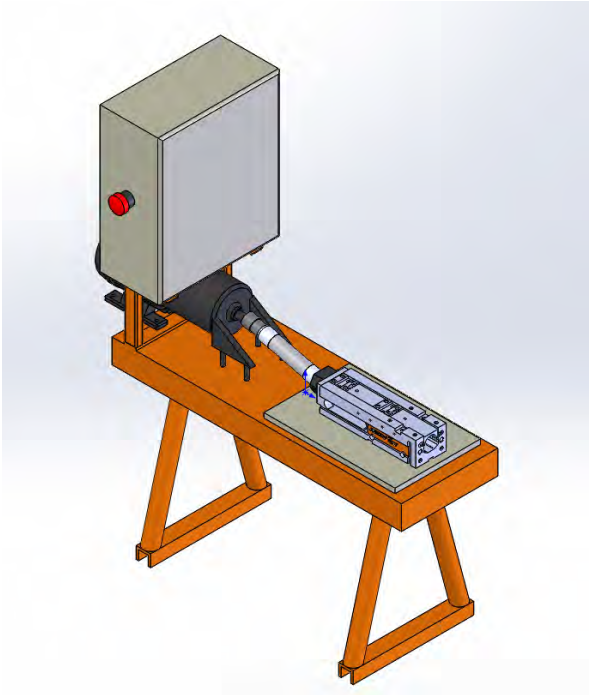


Figura 36. Banco de pruebas
Fuente: Elaboración propia

4.1.1. Transporte

Para transportar o posicionar los viejos módulos, se necesitaba alguna máquina que los pudiera levantar. El nuevo banco de pruebas solo requiere que el usuario desbloquee las ruedas, y lo empuje a la posición deseada. Esto supone un ahorro de tiempo y recursos.

4.1.2. Montaje de las mordazas

Anteriormente era complicado montar el acoplamiento entre motor y mordaza. El nuevo acoplamiento solo requiere posicionar correctamente la mordaza y ajustarla con las tuercas de ajuste. El nuevo banco de pruebas requiere sin embargo, que se defina la altura, como ya se explicó en el manual de uso. Esto implica un trabajo adicional que antes no era necesario, aunque no es muy complicado y genera un mejor resultado que el procedimiento anterior.

4.1.3. Toma de datos

Antes era necesario detener las pruebas, desacoplar la mordaza, instalar un sensor de fuerza, y recién entonces se realizaba la medición de fuerza (figura 37). Después de esto, debía montarse la mordaza de nuevo y se debía poner el sistema en marcha. No se podía medir torque ni ángulo.

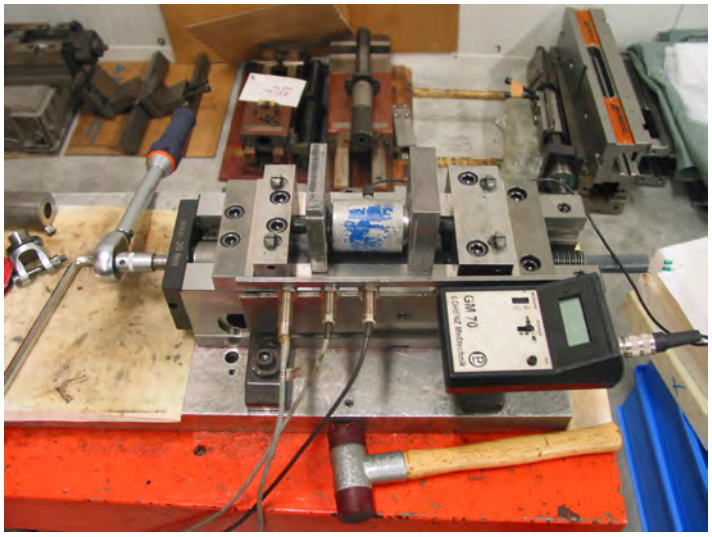


Figura 37. Toma de datos
Fuente: Elaboración propia

El nuevo banco de pruebas graba automáticamente los datos por cada ciclo. Esto supone una mejora significativa en cuanto a la cantidad de datos recolectada, y una disminución drástica en la cantidad de tiempo necesaria para obtener los datos.

La tabla 8 muestra un ejemplo de los datos obtenidos con el nuevo banco de pruebas. Con estos datos se pueden realizar gráficas y analizar los resultados para poder decidir si el diseño fue correcto o si hace falta algún cambio.

Tabla 8: Ejemplo de datos

Fuerza	Momento	Momento en el acople	Posición alcanzada	Posición en el acople	Posición en el desacople	Posición inicial	Modo de control
40.6157	31.383	13.2955	828.4	594.4	566.7	143.5	Kmax_40
40.0121	30.817	13.194	824.5	593.8	566.7	143.5	Kmax_40
40.0079	30.8154	13.1895	824.5	593.8	566.7	143.5	Kmax_40
40.0026	30.7455	13.2106	825.1	593.8	566.7	143.5	Kmax_40
40.0601	30.7882	13.1651	825.1	593.8	566.7	143.5	Kmax_40
39.9138	30.68	13.2473	824.5	593.8	566.7	143.5	Kmax_40
40.0079	30.7	13.0758	825.8	593.8	566.7	143.5	Kmax_40
40.0392	30.7027	13.1041	825.1	593.8	566.7	143.5	Kmax_40
39.9504	30.6539	13.1629	824.5	594.4	566.7	143.5	Kmax_40
40.0549	30.7205	13.1546	825.1	593.8	566.7	143.5	Kmax_40
39.9817	30.6195	13.2056	824.5	594.4	566.7	143.5	Kmax_40
39.9974	30.6955	13.2983	824.5	594.4	566.7	143.5	Kmax_40
40.0758	30.7166	13.2201	825.1	593.8	566.7	143.5	Kmax_40
39.987	30.6567	13.1973	825.1	593.8	566.7	143.5	Kmax_40
40.0045	30.6639	13.3111	825.1	593.8	566.7	143.5	Kmax_40
39.9817	30.6456	13.1923	824.5	593.8	566.7	143.5	Kmax_40
40.0288	30.6794	13.2395	824.5	594.4	566.7	143.5	Kmax_40
39.9129	30.539	13.2511	823.8	594.4	566.7	143.5	Kmax_40

Fuente: Elaboración propia

4.1.4. Configuración

Para configurar los antiguos módulos, era necesario hacerlo empíricamente. El sistema se detenía cierto tiempo después de alcanzar la posición indicada por un sensor inductivo. Este tiempo se calculaba mediante pruebas hasta que el operador decidía que el tiempo era adecuado. Recién entonces se empezaba la prueba. Además, para configurar el tiempo era necesario manipular la caja de conexiones.

En el nuevo banco de pruebas no es necesario manipular la caja de conexiones. Como ya se explicó más arriba, existe una interfaz de usuario que permite manipular el sistema. El sistema es ahora capaz de calcular los parámetros necesarios, por lo que ya no se necesita realizar pruebas y perder tiempo en eso. Y para los casos en los que no se puede calcular dichos parámetros, el usuario puede ver en todo momento los valores de las variables relevantes, es decir, fuerza, momento y posición. De esta forma, la configuración del banco de pruebas es más exacta y permite mejores resultados.

4.1.5. Ergonomía

La altura de los anteriores módulos era muy poca, estaba alrededor de 40 cm. Esto implicaba que el operador debía agacharse para manipular el sistema.

El nuevo módulo tiene una altura conforme a las normas DIN, y permite que cualquier persona lo utilice cómodamente.

4.1.6. Manejo de errores

Los módulos anteriores no contaban con un sistema de detección de errores. Según la experiencia de los trabajadores de la empresa, varias veces se sobrecalentaba el motor, o simplemente se detenía. Tampoco era posible saber cuándo el sistema se había detenido por error. Era necesario entonces, verificar cada cierto tiempo que el banco de pruebas seguía funcionando correctamente, de lo contrario, era posible que el banco de pruebas estuviera detenido, cuando se pensaba que estaba funcionando.

Incluso cuando alguien se daba cuenta de que el sistema había tenido un fallo, no era posible determinar la razón de dicho problema.

El banco de pruebas construido en este proyecto indica al usuario existe un error o problema, y le permite corregirlo. Aparte de eso, el sistema muestra un mensaje indicando el tipo de error, es decir, la causa del problema; esto permite darle una rápida solución.

Otra ventaja que tiene el nuevo módulo es que no se pierde datos cuando sucede un error. Con los viejos módulos, al no saber la causa del error, no se podía saber si los datos tomados eran útiles, o si servía de algo tomar nuevos datos. El fallo podría haber dañado la mordaza, o cambiado su funcionamiento.

4.1.7. Capacidad

El nuevo módulo tiene la capacidad de probar todas las mordazas de la empresa. Los anteriores solo podían producir hasta 40 N m. Los nuevos modelos de mordaza de la empresa llegan a necesitar hasta 100 N m en sus pruebas. El nuevo banco de pruebas es capaz de realizar pruebas incluso si los futuros modelos requieren un mayor torque.

4.1.8. Seguridad

Los modelos anteriores no contaban con ningún mecanismo de seguridad. El nuevo banco de prueba cuenta con diversos elementos de seguridad:

- Relés de seguridad dentro de la caja de conexiones protegen al usuario de corrientes excesivas.
- Cubierta de seguridad impide que el usuario acceda a la zona donde se produce la fuerza de compresión, eliminando cualquier posibilidad de que el operador introduzca su mano entre las partes en movimiento.
- La cubierta de seguridad también protege al usuario en caso de la rotura de una boca, lo cual resulta en pedazos de metal volando a gran velocidad.
- El sistema cuenta con mecanismos de seguridad para prevenir daños en la mordaza. Valores ilógicos son reconocidos; el sistema se detiene ante esta situación.
- El sensor inductivo evita daños en la mordaza. Si la mordaza se mueve más allá de lo debido, el sensor inductivo dejará de mandar la señal correspondiente al sistema y, por lo tanto, se detendrá.

4.2. Análisis de los resultados

A partir de los datos obtenidos en las pruebas se puede elaborar gráficas que permiten analizar el comportamiento de las mordazas. A continuación se muestra las gráficas obtenidas en dos de las pruebas. La primera de estas fue realizada con el modo de control según momento o torque. Como se puede apreciar en la figura 38, el set point es 40 Nm. Se puede apreciar un error en estacionario de 40 N m; para esta aplicación, este error es aceptable.

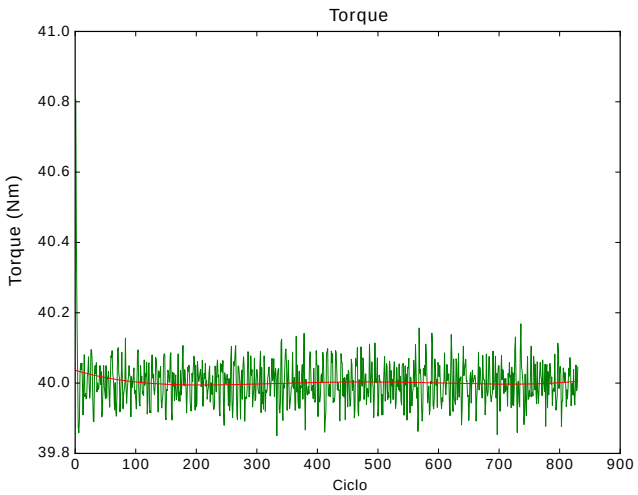


Figura 38. Torque
Fuente: Elaboración propia

Los otros parámetros relevantes de la prueba son los siguientes:

- Duración de la posición cerrada: 10 s

- Duración de la posición abierta: 10 s
- Apertura de la mordaza: 3 mm

Ya que el torque va a ser constante, los parámetros que interesa medir son la fuerza y la posición. La fuerza alcanzada se muestra en la figura 39. Como es de esperarse, este valor disminuye con el uso, debido al desgaste de la mordaza.

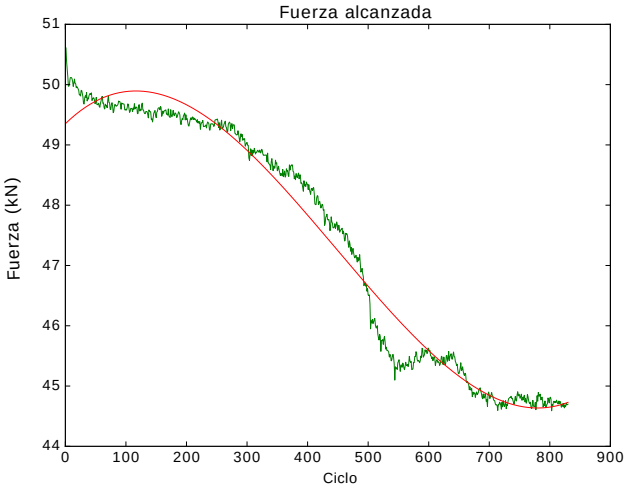


Figura 39. Fuerza
Fuente: Elaboración propia

La posición alcanzada se muestra en la figura 40. Se puede apreciar que este valor también disminuye. El desgaste de la mordaza hace que sea necesario un mayor torque de entrada para conseguir una misma salida, lo que causa que se alcance el valor de torque deseado, antes de llegar a la posición óptima. Si bien la variación en la posición es apenas apreciable –5° aproximadamente–, se puede esperar que, al realizar más ciclos, esta variación se vuelva considerable.

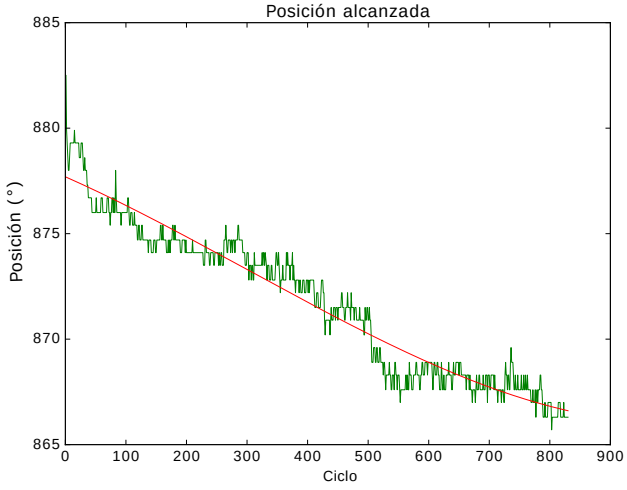


Figura 40. Posición
Fuente: Elaboración propia

En la figura 41 se aprecia que el torque necesario para acoplar el multiplicador de fuerza va disminuyendo con el uso. Al final de la gráfica se aprecia un aumento de este valor.

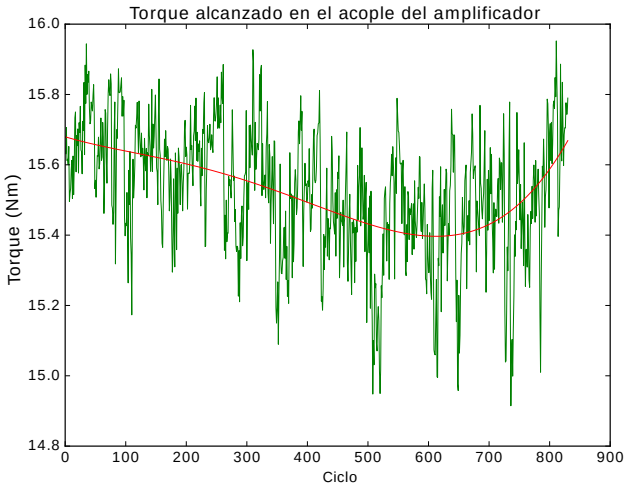


Figura 41. Torque en el acople
Fuente: Elaboración propia

La disminución de este torque significa que la fuerza multiplicada en el amplificador de fuerza es menor.

La siguiente prueba que se muestra fue realizada con el modo de control según fuerza. Los otros parámetros relevantes fueron iguales que en la anterior. La fuerza objetivo fue establecida en 40 kN, como se puede apreciar en la figura 42. El error en este caso también puede ser considerado como despreciable.

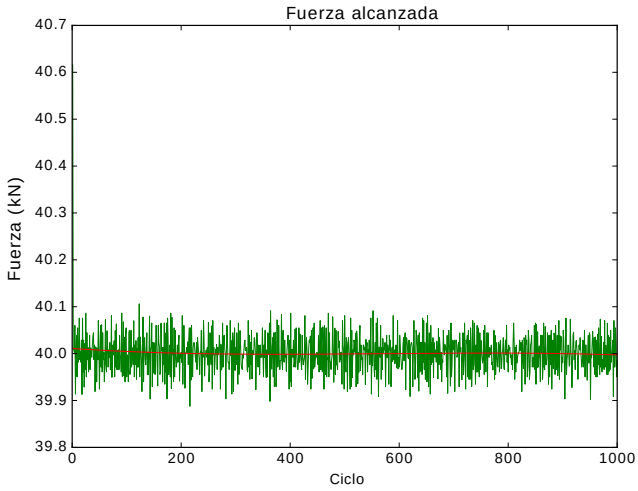


Figura 42. Fuerza
Fuente: Elaboración propia

El torque obtenido se muestra en la figura 43. En este caso se observa que el momento necesario para conseguir una misma fuerza aumenta con el uso.

La posición alcanzada sigue la misma tendencia que en la anterior prueba, disminuyendo con el desgaste de la mordaza, como se muestra en la figura 44. Esto confirma el hecho

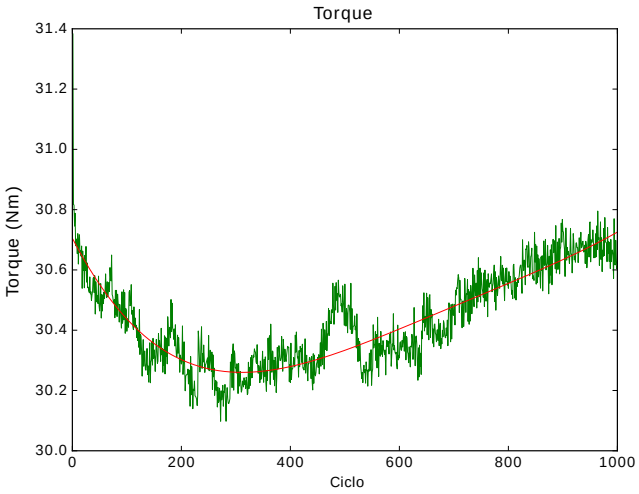


Figura 43. Torque
Fuente: Elaboración propia

de que, a medida que se utiliza la mordaza, la resistencia del sistema ante el uso aumenta, por lo que se llega a necesitar un mayor torque para llegar a una menor posición.

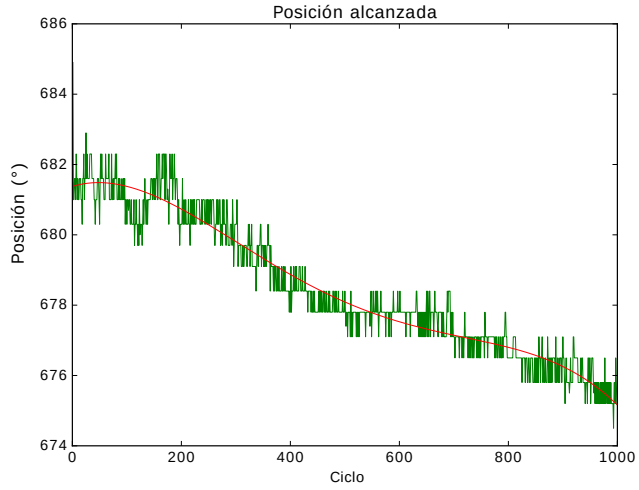


Figura 44. Posición
Fuente: Elaboración propia

Por último, tenemos el torque necesario para el acople del amplificador de fuerza, mostrado en la figura 45. El comportamiento en este caso es más caótico, pero mantiene la tendencia de reducirse con el uso.

Estas fueron unas de las primeras pruebas realizadas con el banco de pruebas. Después de un análisis más completo de los resultados, se determinó que el tiempo de las posiciones cerrada y abierta era muy corto, y por eso se daba un desgaste tan significativo. Modificaciones a estos parámetros fueron realizadas para pruebas posteriores.

Estos gráficos permiten no solo estimar el tiempo de vida de la mordaza, sino también encontrar posibles mejoras en el diseño. Por ejemplo, el comportamiento del momento en el amplificador de fuerza implica que los resortes utilizados para el acoplamiento del amplificador se han gastado después del número de ciclos empleado. Si la aplicación real de la

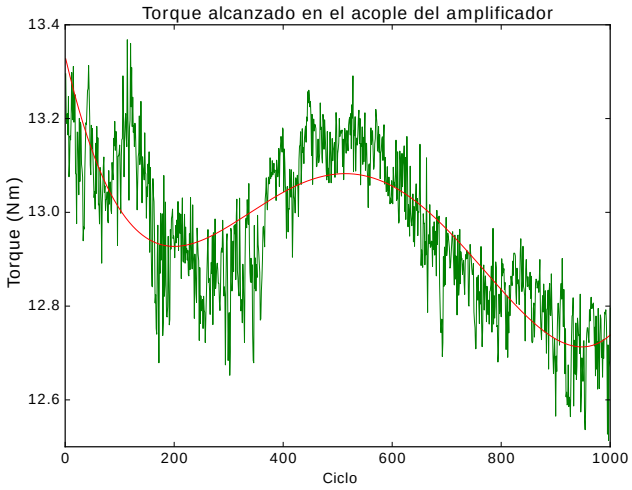


Figura 45. Torque en el acople
Fuente: Elaboración propia

mordaza supera este número de ciclos, se podría utilizar resortes más resistentes; o, en todo caso, prevenir al cliente que, después de dicho número de ciclos, es necesario realizar un mantenimiento de la mordaza.

Como es de esperarse, una sola prueba no es suficiente para obtener conclusiones definitivas. Después de una prueba, se ajustan los parámetros deseados, se hacen cambios en el diseño, o se prueban nuevos elementos; para después realizar nuevas pruebas, obtener nuevos resultados, y llegar al mejor diseño posible.

Capítulo 5
Conclusiones y recomendaciones

El nuevo banco de pruebas presenta una mejora considerable en comparación con los módulos previos. El proyecto desarrollado permite ahora realizar pruebas más completas, y en menor tiempo. El banco de pruebas actual también cuenta con mecanismos para controlarlo y monitorear las pruebas a distancia. En general, se puede decir que los objetivos planteados han sido cumplidos.

5.1. Seguridad y complejidad del sistema

El nuevo sistema incluye nuevos procesos y tiene mayores capacidades, lo cual aumenta su complejidad. Si bien las nuevas capacidades y precauciones significan una mejora, cada nueva rutina implementada implica nuevas posibilidades de error, por lo que debe implementarse mecanismos de seguridad para cada una. Aparte de eso, al tener el banco de pruebas una capacidad mayor en cuanto al torque disponible, los errores son más críticos.

Es necesario también tener en cuenta que, al programar los mecanismos de seguridad, es posible influenciar otros procesos sin querer. Por ejemplo, si se reconoce que, en un punto, el torque aumenta demasiado rápido, es posible que haya algún bloqueo en la rosca. Sin embargo, también existe un pico de torque cuando se acopla el amplificador de fuerza, o cuando hay un tope de momento; estas situaciones no implican un error, por lo que el sistema no debe parar en ellas. Se necesita entonces, implementar una rutina en el programa que sea capaz de diferenciar las situaciones normales de la situación de error, y actuar de acuerdo a eso.

5.2. Problemas ocurridos

Durante el desarrollo de este proyecto ocurrieron problemas que permitieron comprobar la relevancia de ciertos conceptos explicados en capítulos previos; a continuación se explica estos casos.

5.2.1. Tolerancias

Las tolerancias son, definitivamente, de suma importancia. No especificar su valor, o establecer este último equivocadamente, puede llevar a problemas en el proceso de construcción y en el funcionamiento del sistema en general. Por ejemplo, la extensión requerida para el sistema de husillo del plato de altura variable, está soportada por un rodamiento para facilitar su rotación. Sin embargo, la tolerancia del diámetro, debido a un descuido, no fue especificada. Para toda medida no tolerada se utiliza un valor de tolerancia preestablecido que depende de la magnitud de la dimensión. En este caso, la tolerancia requerida permite la variación del valor real sin llegar nunca a ser mayor que el de diseño. Por el contrario, la tolerancia predeterminada para este valor de diámetro permite hasta 0.5 mm más que el valor de diseño. Debido a esto, la pieza recibida no podía entrar en el rodamiento, y, por lo tanto, se requirió trabajo extra para llegar al diámetro óptimo. Si bien este problema no fue crítico, significó una pérdida de tiempo que podría haber sido evitada.

5.2.2. Ajustes

La importancia de los ajustes también pudo ser observada, más que nada, en la impresión de piezas en 3D. Para algunas partes en las que se necesitaba una forma geométrica complicada, y no era necesario tener una resistencia mecánica tan elevada, se optó por la impresión 3D. En este caso, las dimensiones finales suelen ser menores a las del modelo 3D debido a la contracción del material al enfriarse la pieza. Por eso es necesario sobredimensionar el modelo antes de imprimirlo.

También se observó la influencia de los ajustes en otros elementos. Las dimensiones de los elementos en un buen diseño debe asegurar que su función sea correctamente desempeñada, sin que la pieza resulte muy costosa. En el caso del sistema de rosca trapezoidal, se escogió una rosca que, en principio, debería haber sido suficiente, pero trabajaría en su límite. Debido a diversos factores durante el proceso de montaje, esta rosca resultó ser muy corta, por lo que se tuvo que comprar una nueva más larga. Este es uno de los casos en los que un sobredimensionamiento podría haber evitado problemas.

5.3. Recomendaciones

Como es de esperar, ningún trabajo es perfecto. Si bien se ha cumplido con los objetivos planteados, queda espacio para futuras modificaciones; las posibilidades de mejora encontradas al momento de dar por finalizado mi proyecto de tesis se nombran a continuación:

- La cubierta de seguridad tiene unas tiras que impiden que se abra excesivamente. Esto no es muy estable, por lo que sería bueno encontrar una mejor forma de sujetarla. Aparte de eso, el material de la cubierta no es muy rígido, por lo que es difícil maniobrarla. Sería conveniente entonces, implementar algún tipo de estructura de soporte o de refuerzo para la cubierta, para solucionar el problema de la maniobrabilidad.

- El macho en el cual se encaja el dado para conectar el motor con la mordaza es de un material no muy resistente. Durante las pruebas, este elemento se rompió, por lo que es muy recomendable seleccionar otro material, o cambiar el diseño para aumentar la resistencia de la pieza.

- El servidor VNC es bastante lento. Se puede utilizar, pero sería bueno encontrar una mejor forma de acceder a la HMI.
- La placa de altura regulable para las mordazas tiene un rango limitado de alturas, pero no hay ningún mecanismo que impida al usuario intentar variar la altura más allá de sus límites. Esto último causaría la falla del sistema de husillo utilizado para este elemento. Siguiendo con este elemento, si el usuario intenta variar la altura cuando los frenos están aplicados, es muy probable que en este caso también se malogre el husillo. Se recomienda entonces, implementar un mecanismo que impida al usuario llegar a estas situaciones, o poner una advertencia para que esto no suceda.

Por último solo me queda decir que el módulo se encuentra operativo y se usa frecuentemente. Hay algunos problemas como es de esperarse de cualquier sistema nuevo, pero eso no impide su utilización, y se puede llevar a cabo las pruebas y obtener los resultados correspondientes.

Bibliografía

[1] Automation (22 de octubre del 2016). Recuperado el 25 de octubre de 2016, de Wikipedia: https://en.wikipedia.org/wiki/Automation

[2] Hayden, H., Assante, M. & Conway, T., (agosto 2014). *An Abbreviated History of Automation and Industrial Control Systems and Cybersecurity*, p. 12

[3] Vise (2 de agosto de 2016). Recuperado el 25 de octubre de 2016, de Wikipedia: https://en.wikipedia.org/wiki/Vise

[4] Leadscrew (15 de octubre de 2016). Recuperado el 26 de octubre de 2016, de Wikipedia: https://en.wikipedia.org/wiki/Leadscrew

[5] Vises (7 de enero de 2012). Recuperado el 26 de octubre de 2016, de Appropedia: http://www.appropedia.org/Vises

[6] Vahid-Araghi, O. & Golnaraghi, F. (2011) *Friction-Induced Vibration in Lead Screw Drives*. New York: Springer International Publishing AG

[7] Servomotor(4 de diciembre de 2016). Recuperado el 25 de enero de 2017, de Wikipedia: https://en.wikipedia.org/wiki/Servomotor

[8] Eitel, Elisabeth. *Basics of rotary encoders: Overview and new technologies* (7 de mayo de 2014). Recuperado el 25 de enero de 2017, de Machine Design Magazine.

[9] http://electronics.stackexchange.com/questions/15481/how-does-a-ball-mouse-know-the-direction, recuperado el 25.01.17

[10] AE3145 Resistance Strain Gage Circuits

[11] How many different types of force transducer are there? (25 de marzo de 2010). Recuperado el 20 de enero de 2017 de: http://www.npl.co.uk/reference/faqs/how-many-different-types-of-force-transducer-are-there-(faq-force)

[12] 1997 Power Transmission Design

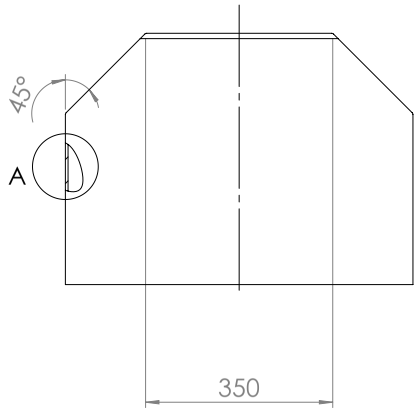
[13] EML2322L – MAE Design and Manufacturing Laboratory

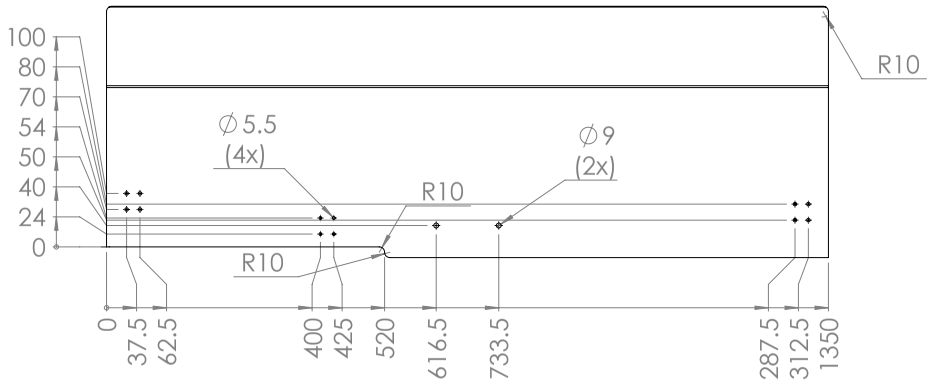
[14] Allowance_(engineering)(14 de noviembre de 2016). Recuperado el 14 de febrero de 2017, de Wikipedia: https://en.wikipedia.org/wiki/Allowance_(engineering)

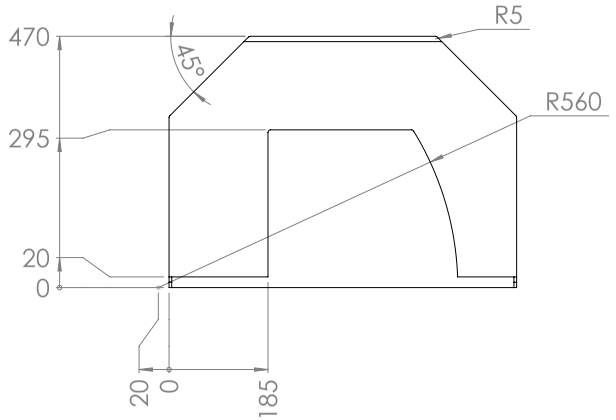
[15] Glassen, Ken, (2013). *Tight Tolerance Is Critical For Top Quality And Superior Performance*. Recuperado de http://www.manufacturing.net/article/2013/11/tight-tolerance-critical-top-quality-and-superior-performance

Anexos

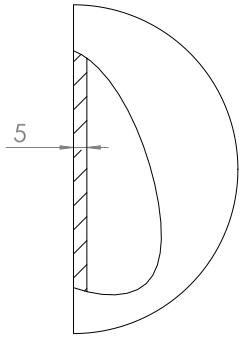
Apéndice A
Planos



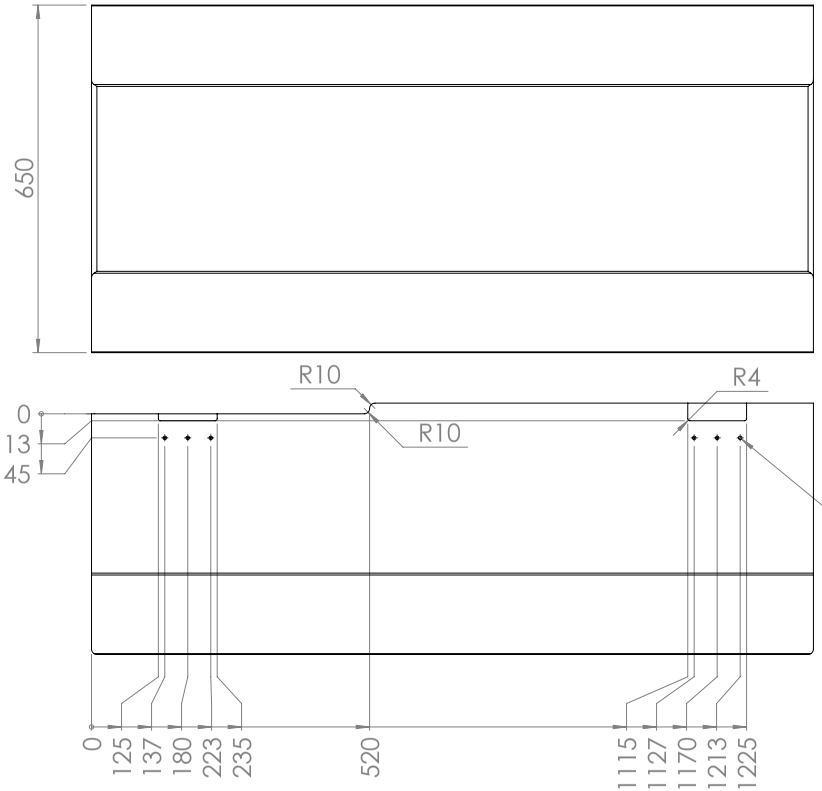


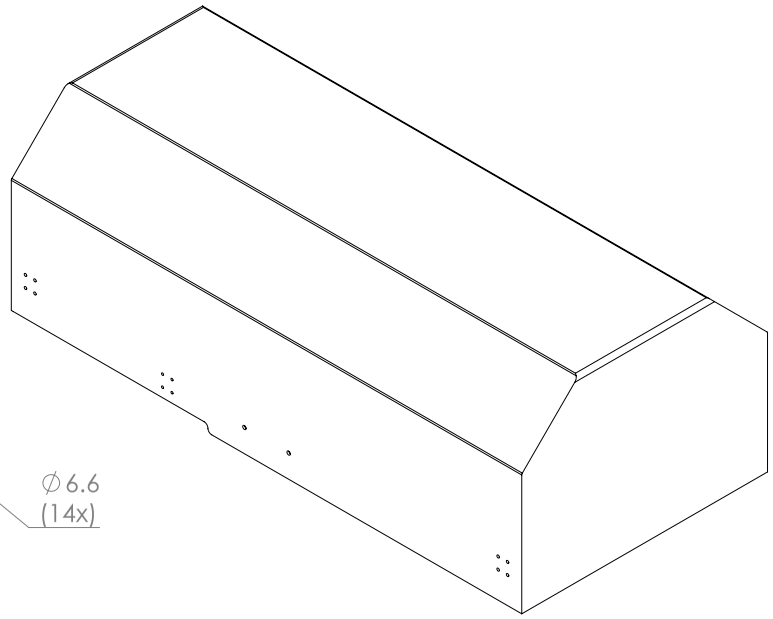


72



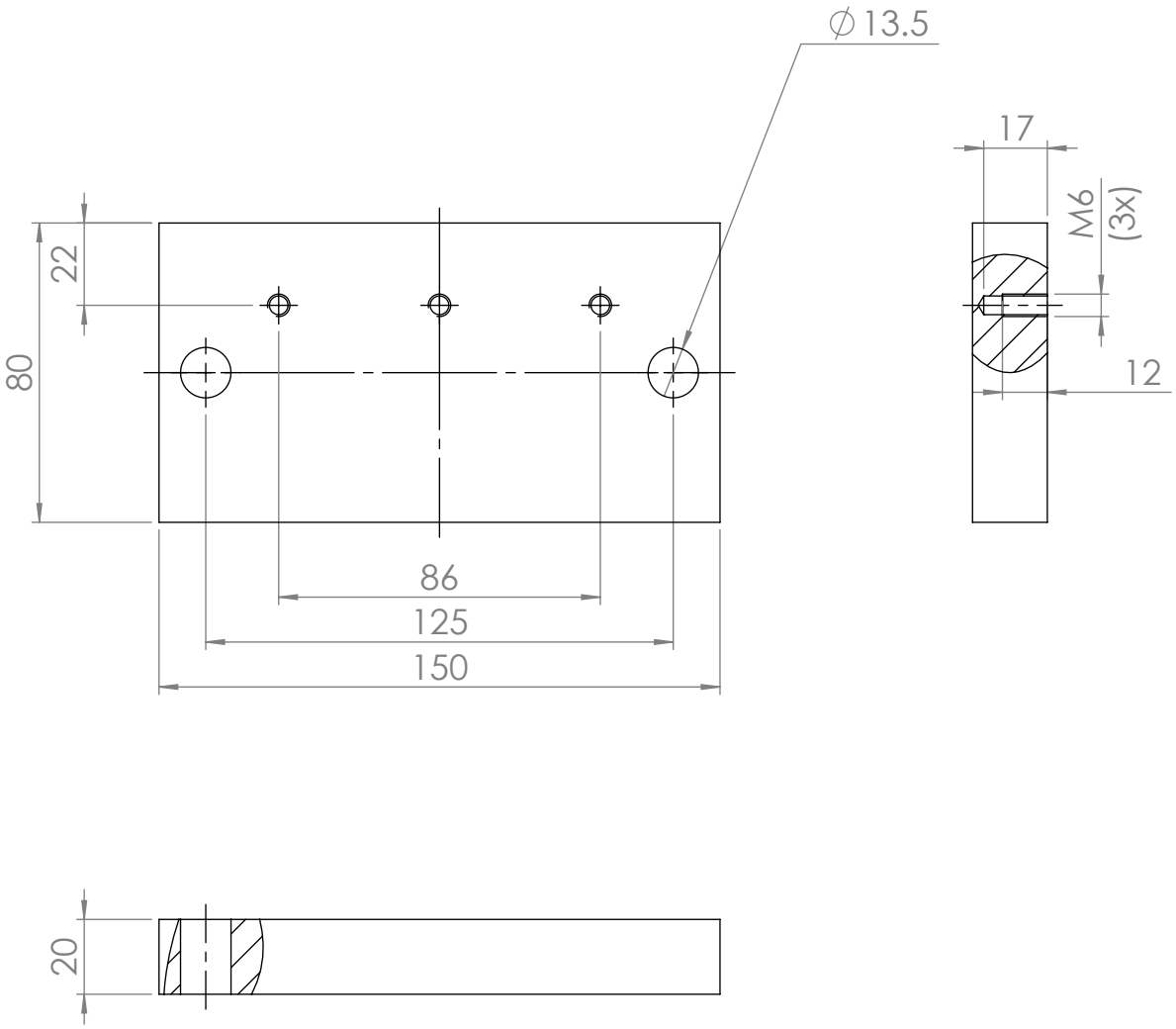
DETALLE A
ESCALA 1 : 2





agujeros sin tolerancia ±0,2

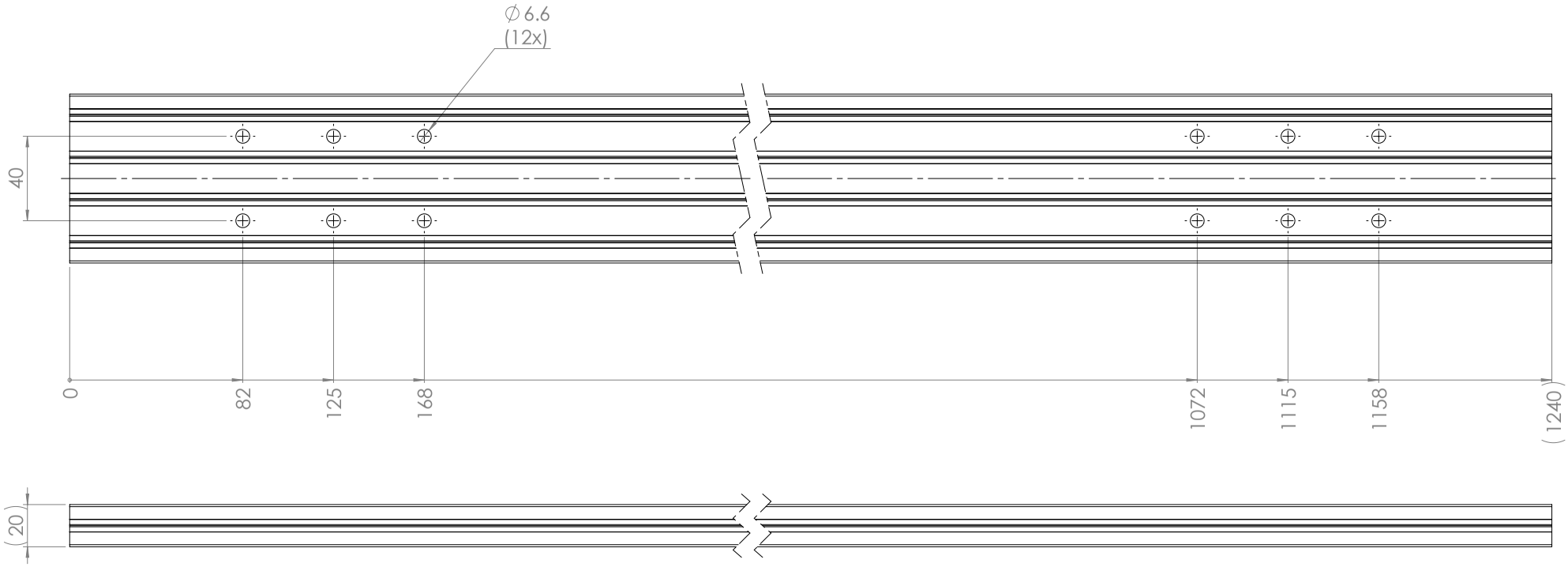
	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	13.02.17			
REVISADO				
SK 43533-001			CUBIERTA	1:10
JOSE FRANCISCO CHUMAN ALVARADO				



agujeros sin tolerancia ±0,2

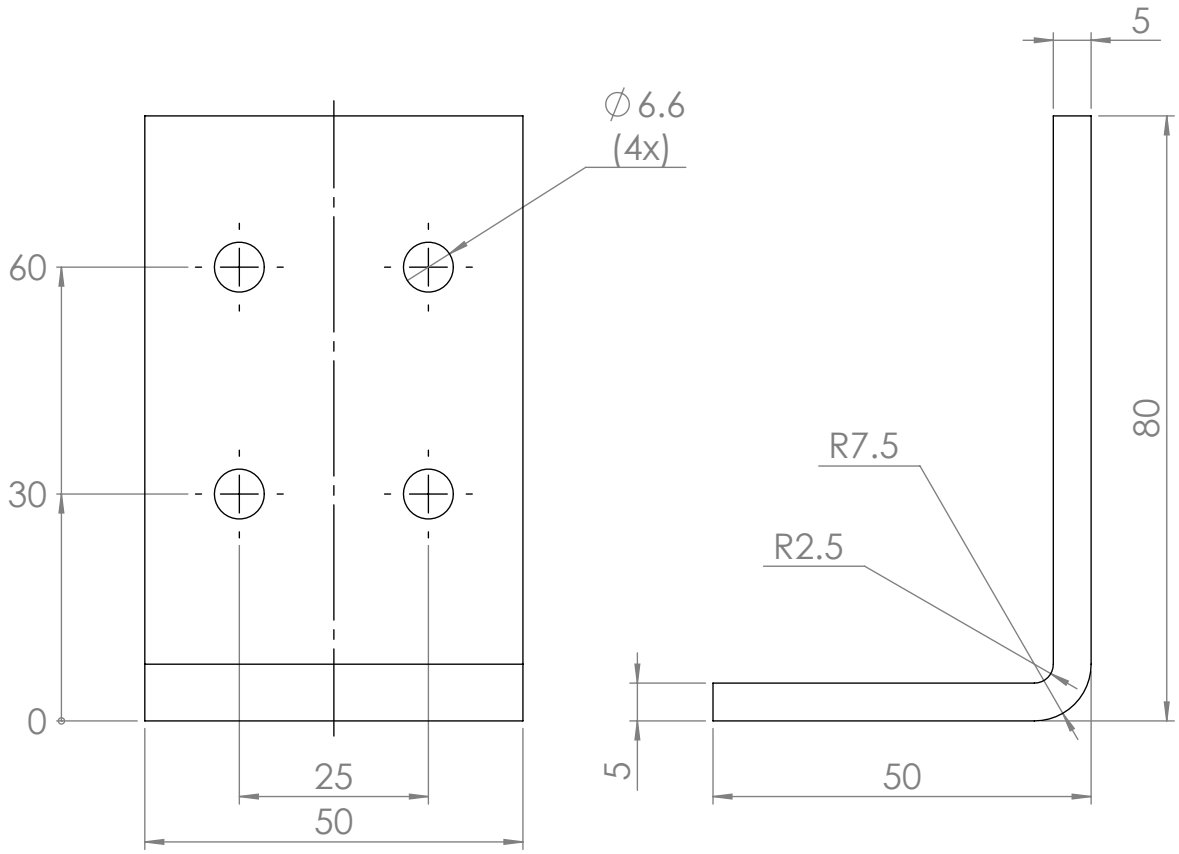
pavonado

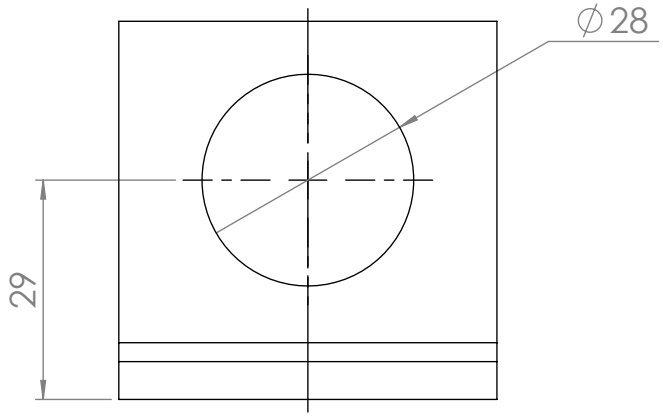
	FECHA	NOMBRE	
DIBUJADO	09.02.17		***UNIVERSIDAD DE PIURA***
REVISADO			*FACULTAD DE INGENIERIA*
SK 43533-002			CONECTOR CUBIERTA — 1:2
JOSE FRANCISCO CHUMAN ALVARADO			



agujeros sin tolerancia ±0,2

	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	09.02.17			
REVISADO				
SK 43533-003			SUJECIÓN PARA LA CUBIERTA	1:2
JOSE FRANCISCO CHUMAN ALVARADO				

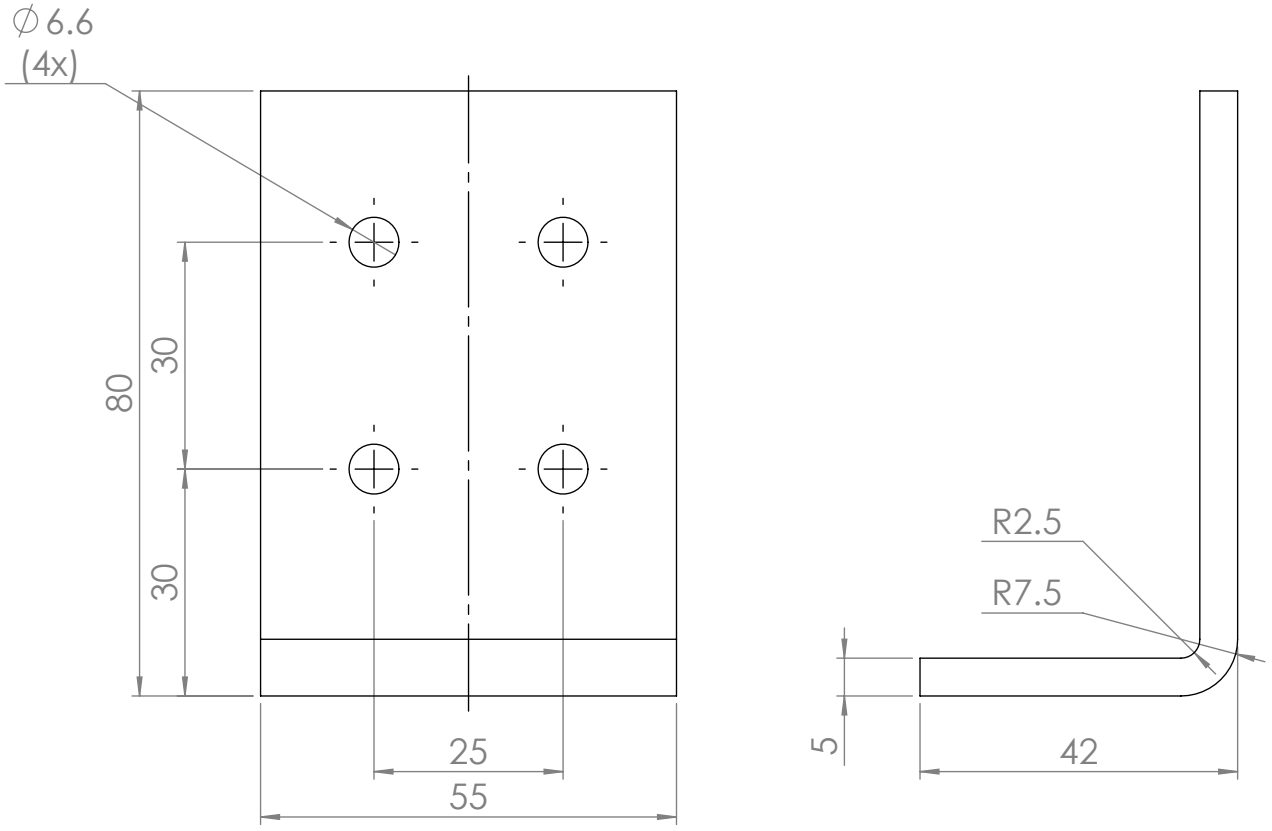


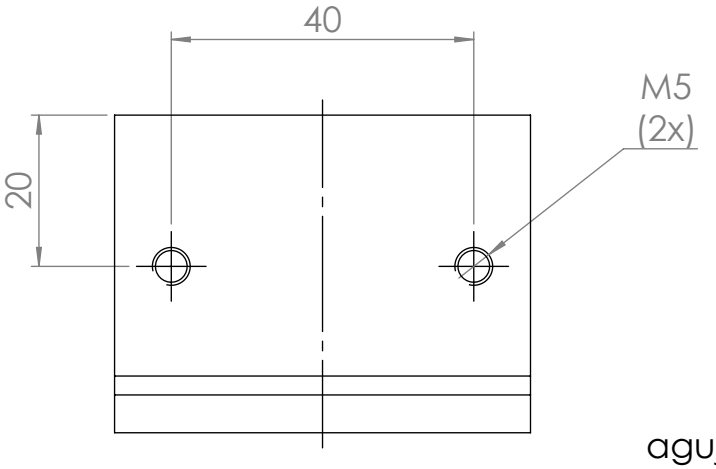


agujeros sin tolerancia ±0,2

pavonado

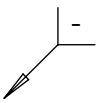
	FECHA	NOMBRE	
DIBUJADO	09.02.17		***UNIVERSIDAD DE PIURA***
REVISADO			*FACULTAD DE INGENIERIA*
SK 43533-004			SUJECIÓN DEL INTERRUPTOR DE SEGURIDAD — 1:1
JOSE FRANCISCO CHUMAN ALVARADO			



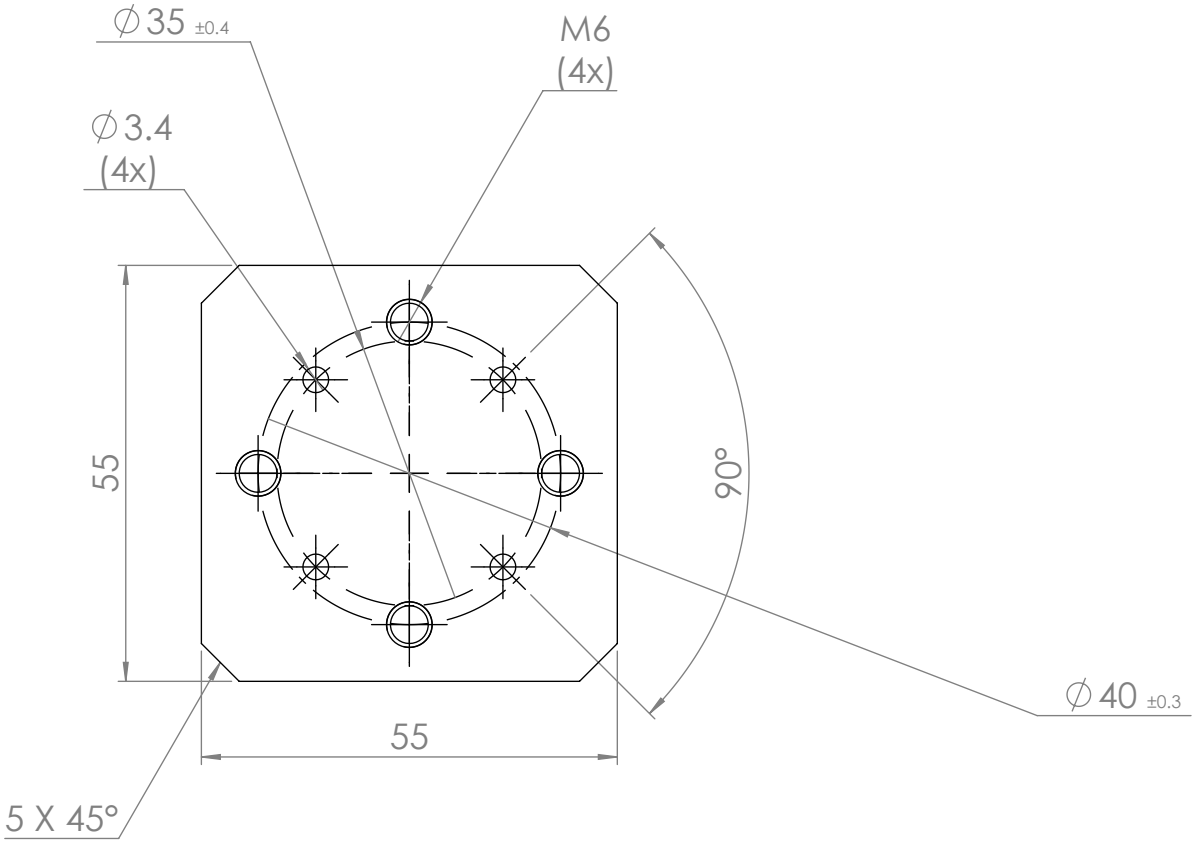


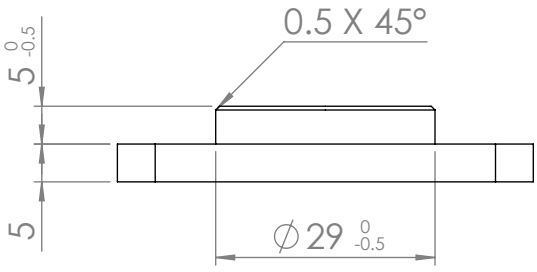
pavonado

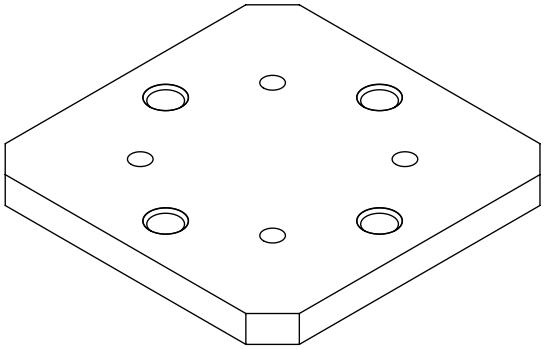
agujeros sin tolerancia ±0,2

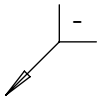


	FECHA	NOMBRE	
DIBUJADO	06.03.17		***UNIVERSIDAD DE PIURA***
REVISADO			*FACULTAD DE INGENIERIA*
SK 43533-007			SUJECIÓN AMORTIGUADORES — 1:1
JOSE FRANCISCO CHUMAN ALVARADO			

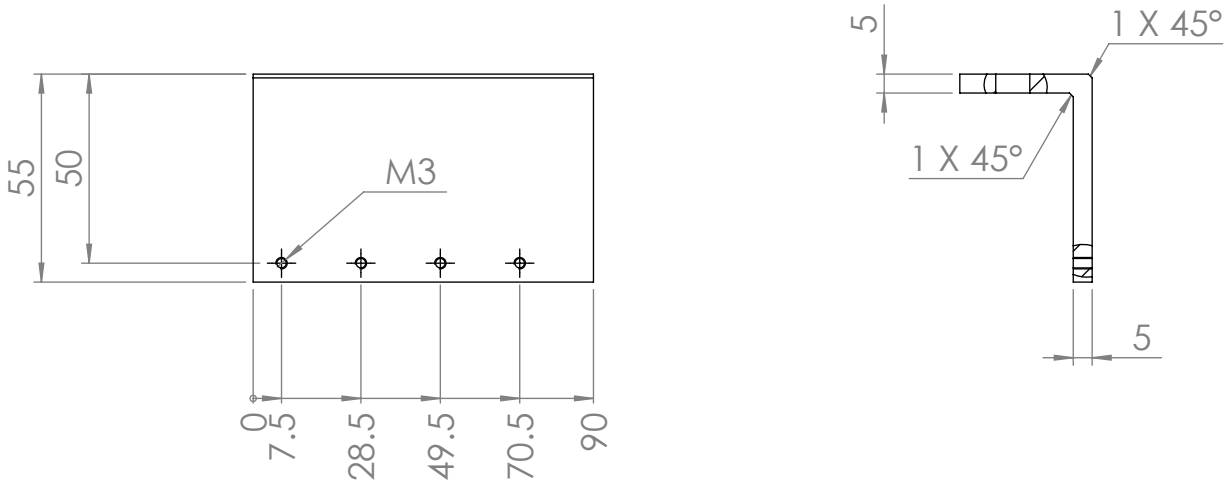


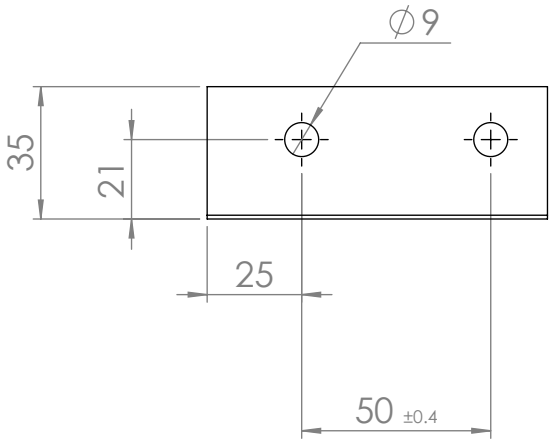


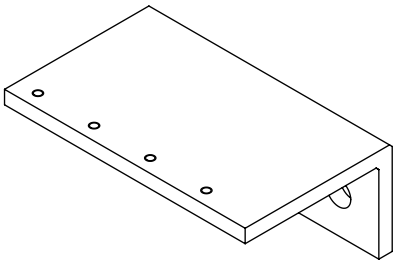


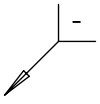


	FECHA	NOMBRE	
DIBUJADO	20.03.17		***UNIVERSIDAD DE PIURA***
REVISADO			*FACULTAD DE INGENIERIA*
SK 43533-010			1:1
JOSE FRANCISCO CHUMAN ALVARADO			CONECTOR

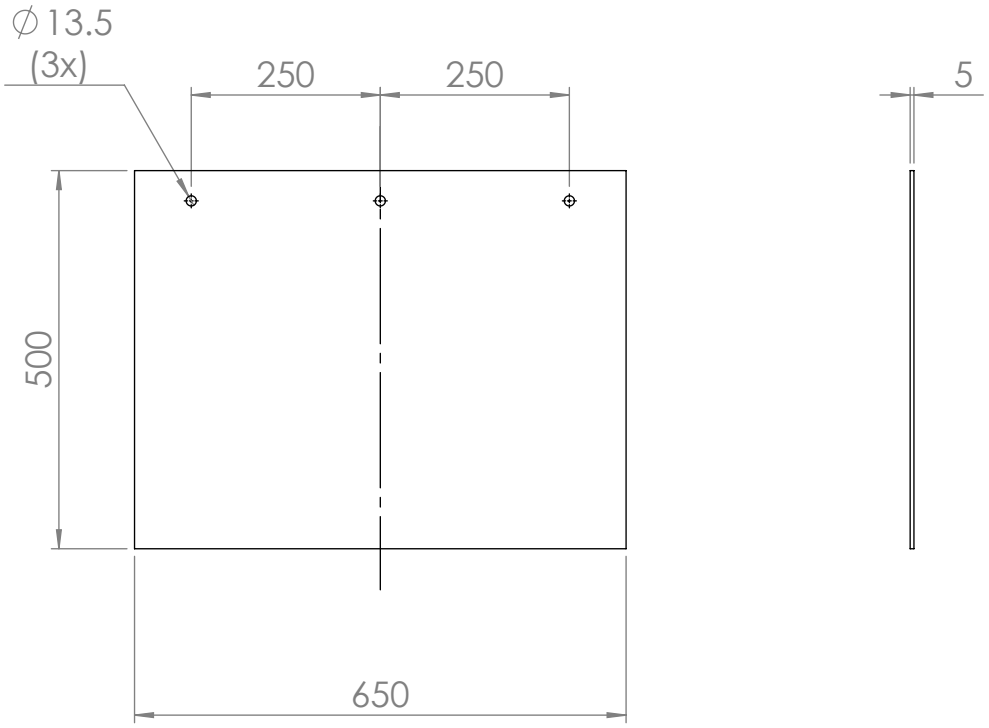


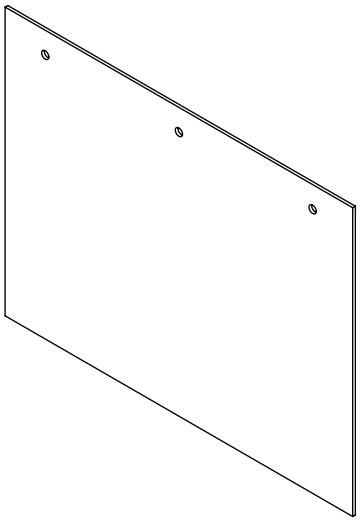






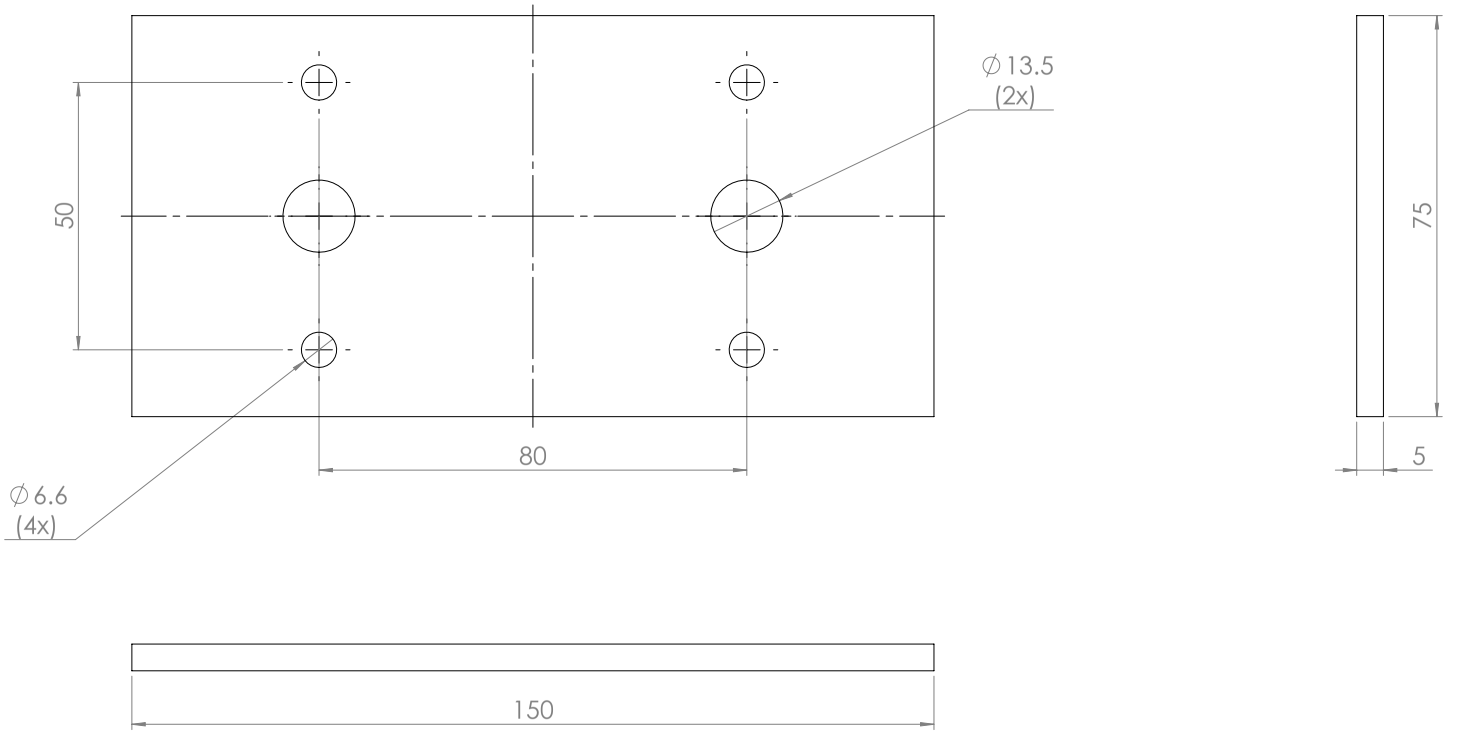
	FECHA	NOMBRE	
DIBUJADO	08.05.17		***UNIVERSIDAD DE PIURA***
REVISADO			*FACULTAD DE INGENIERIA*
SK 43533-011			SUJECIÓN SENSOR INDUCTIVO / 1:2
JOSE FRANCISCO CHUMAN ALVARADO			



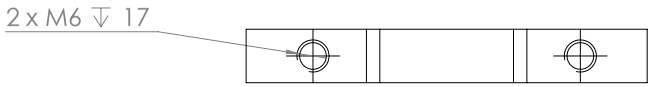


	FECHA	NOMBRE	
DIBUJADO	22.05.17		**UNIVERSIDAD DE PIURA**
REVISADO			FACULTAD DE INGENIERIA

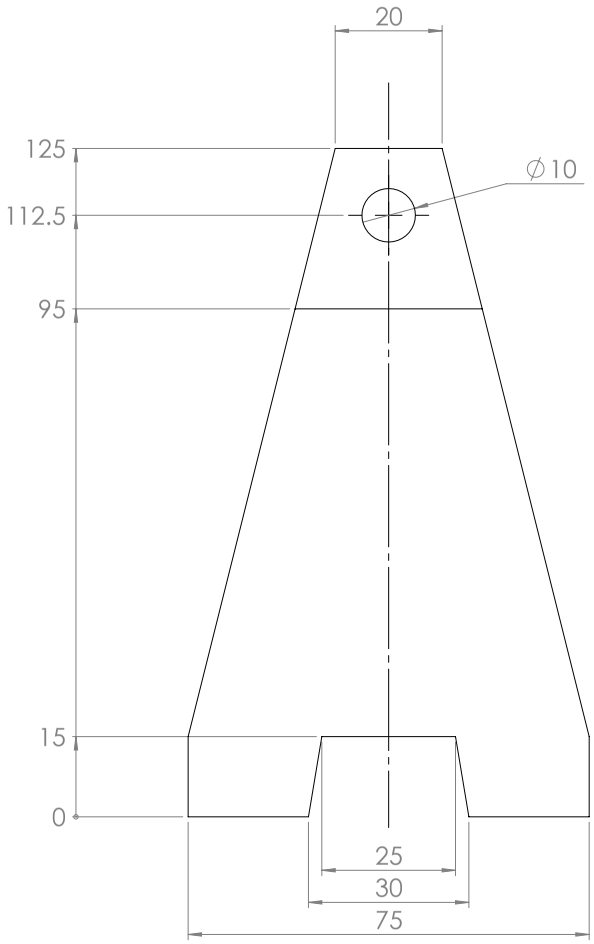
SK 43533-012	PLACA DE SEGURIDAD	1:10
JOSE FRANCISCO CHUMAN ALVARADO		

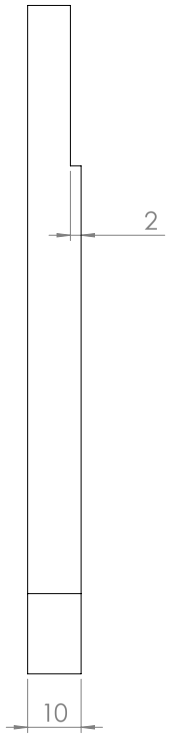


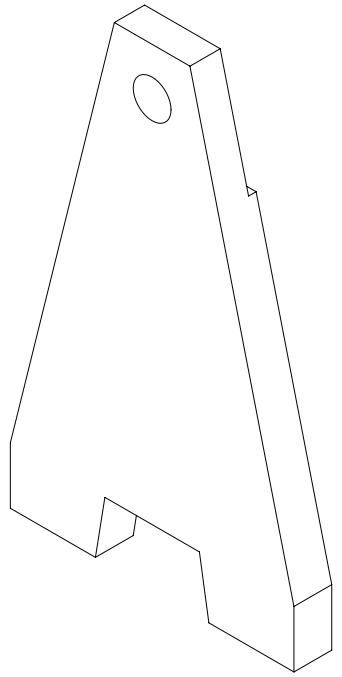
	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	22.05.17			
REVISADO				
SK 43533-013			SUJECIÓN CADENA	1:1
JOSE FRANCISCO CHUMAN ALVARADO				



2X

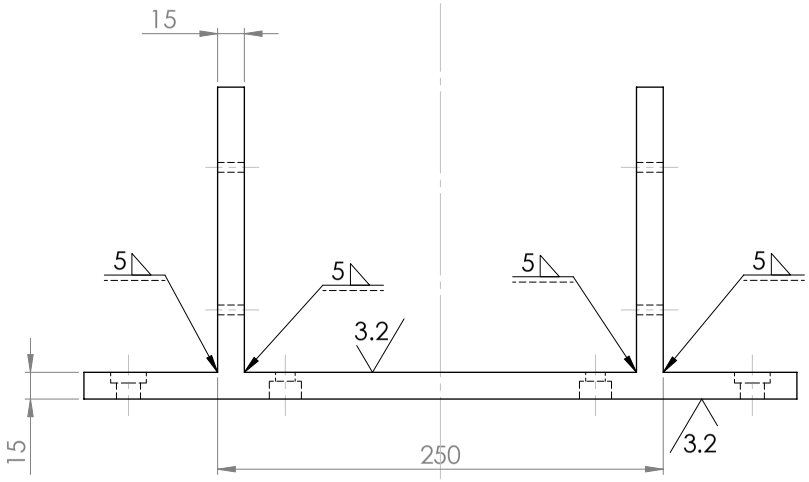


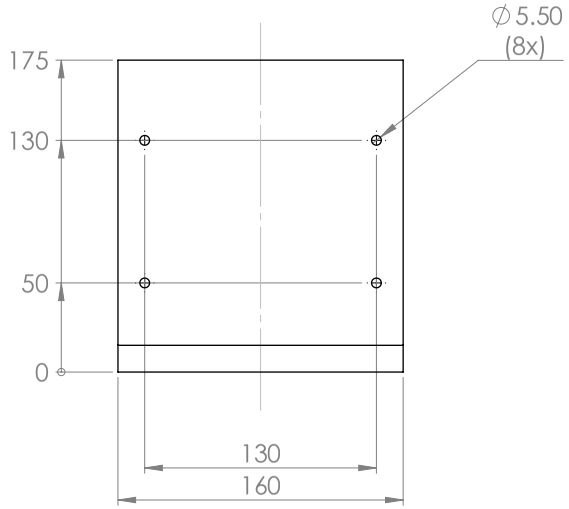


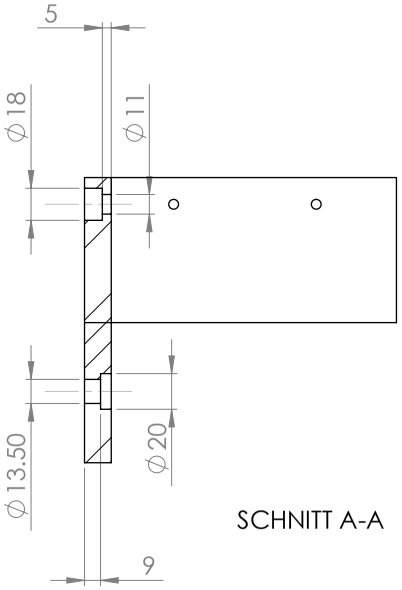


	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	06.03.17			
REVISADO				
SK 43533-014			SUJECIÓN CADENA	1:1
JOSE FRANCISCO CHUMAN ALVARADO				

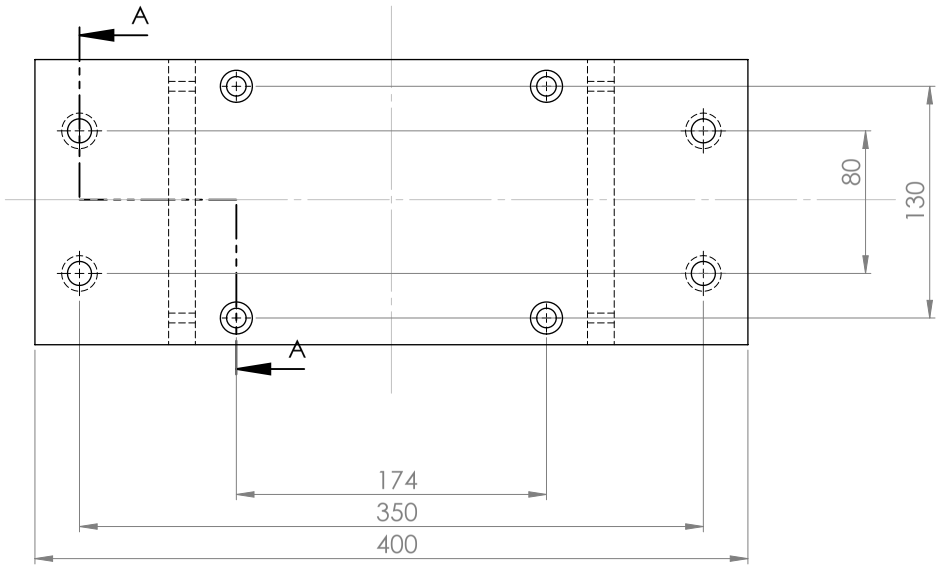
81

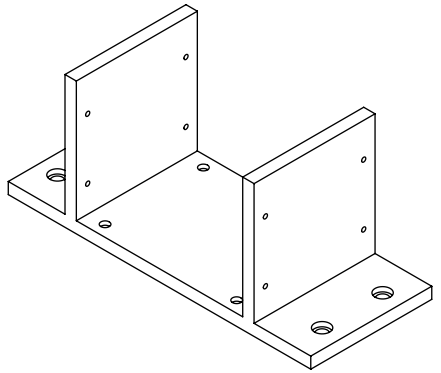






SCHNITT A-A

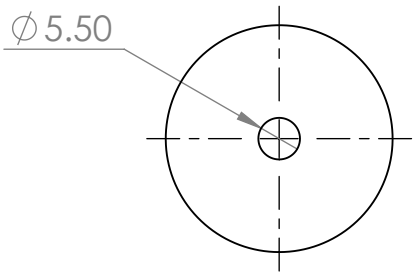


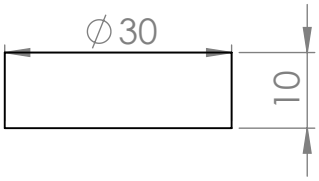


agujeros sin tolerancia ±0,2

3.2 ( ) pavonado

	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	20.12.16			
REVISADO				
SK 43533-019			BASE DE LAS COLUMNAS	1:3
JOSE FRANCISCO CHUMAN ALVARADO				



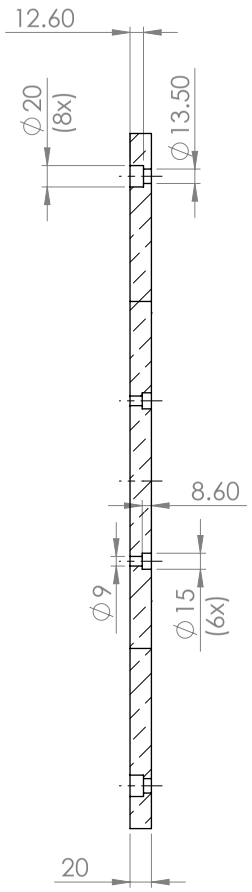


pavonado

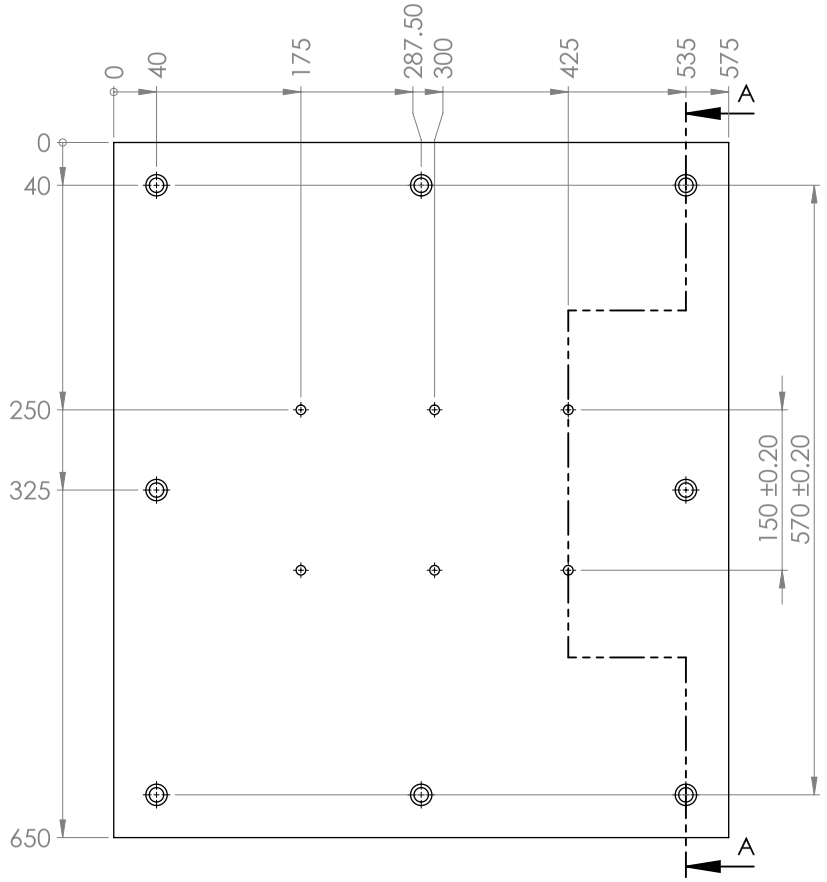
3.2

	FECHA	NOMBRE	
DIBUJADO	03.01.17		***UNIVERSIDAD DE PIURA***
REVISADO			*FACULTAD DE INGENIERIA*

SK 43533-044	ARANDELA	1:1
JOSE FRANCISCO CHUMAN ALVARADO		



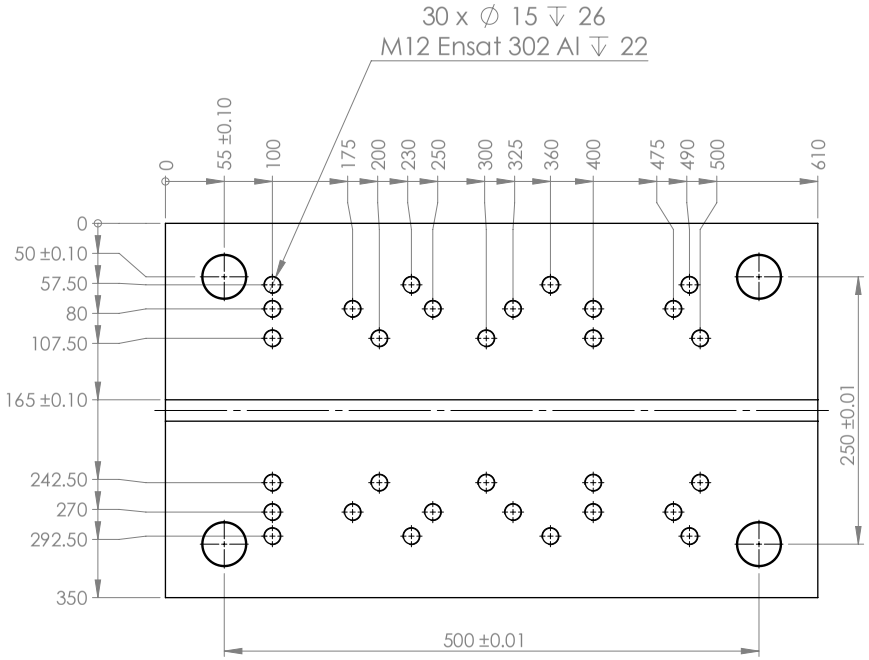
SCHNITT A-A

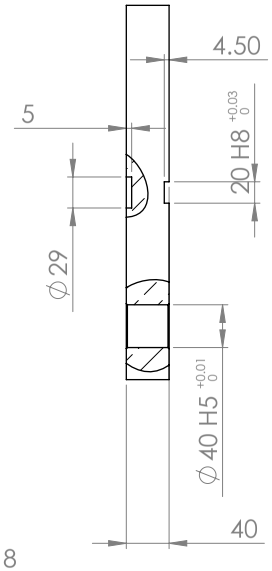


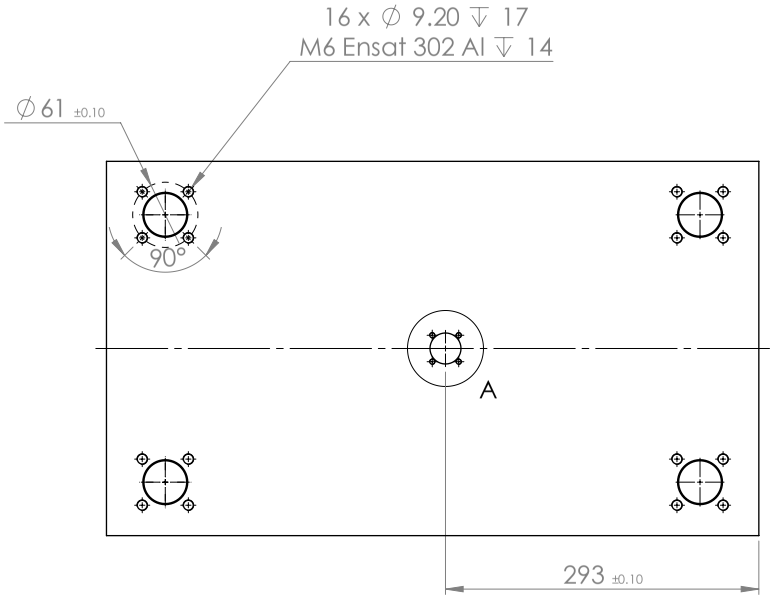
agujeros sin tolerancia ±0,2

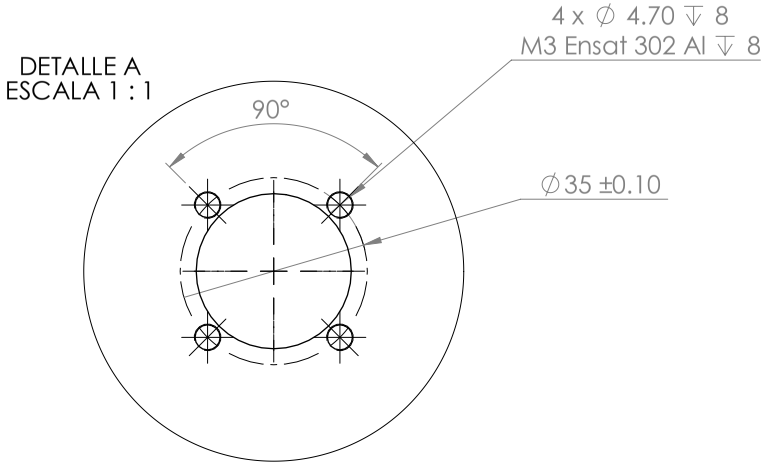
2,2 -0,6

	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	20.12.16			
REVISADO				
SK 43533-053			SUPERFICIE DE MESA - MOTOR	1:5
JOSE FRANCISCO CHUMAN ALVARADO				









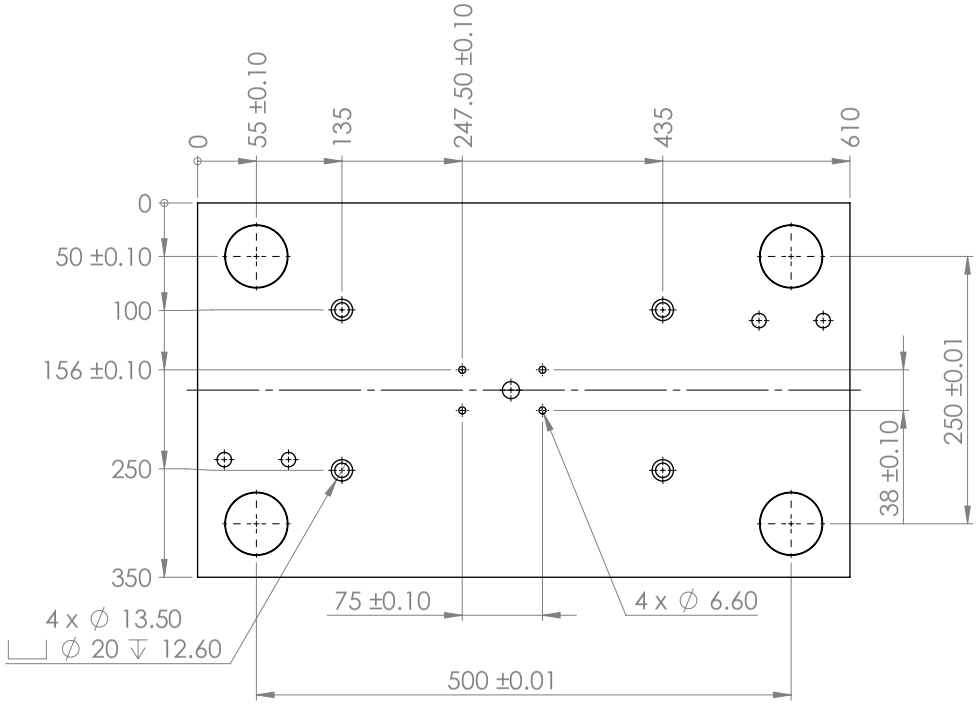
agujeros sin tolerancia ±0,2

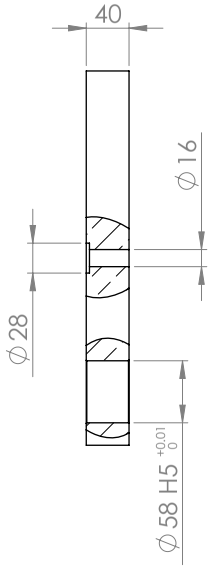
2,2

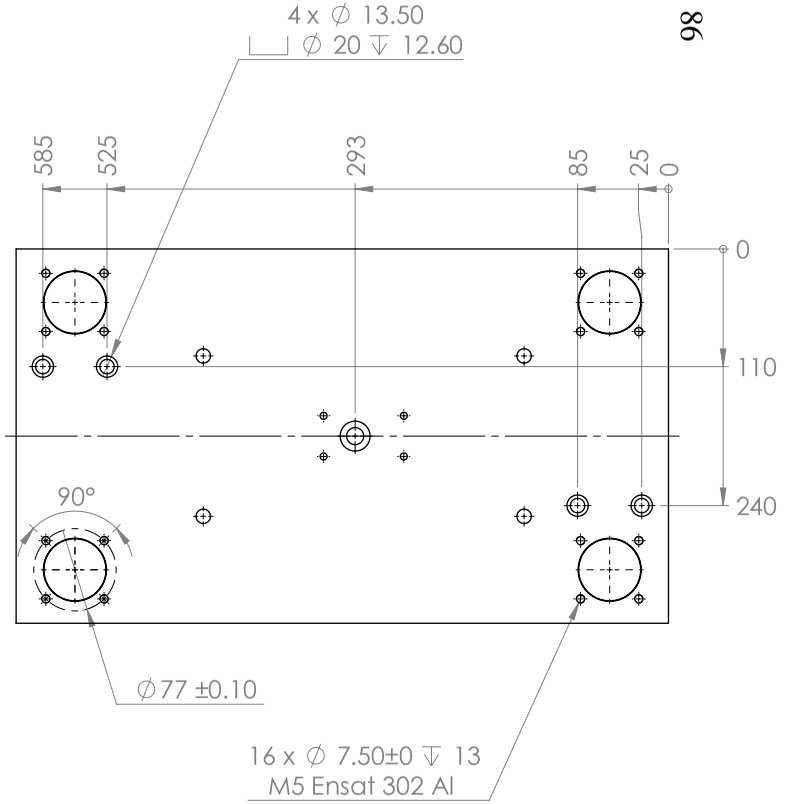
-0,6

	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	21.12.16			
REVISADO				
SK 43533-056			PLATO ALTURA VARIABLE	1:5
JOSE FRANCISCO CHUMAN ALVARADO				

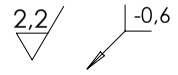
85



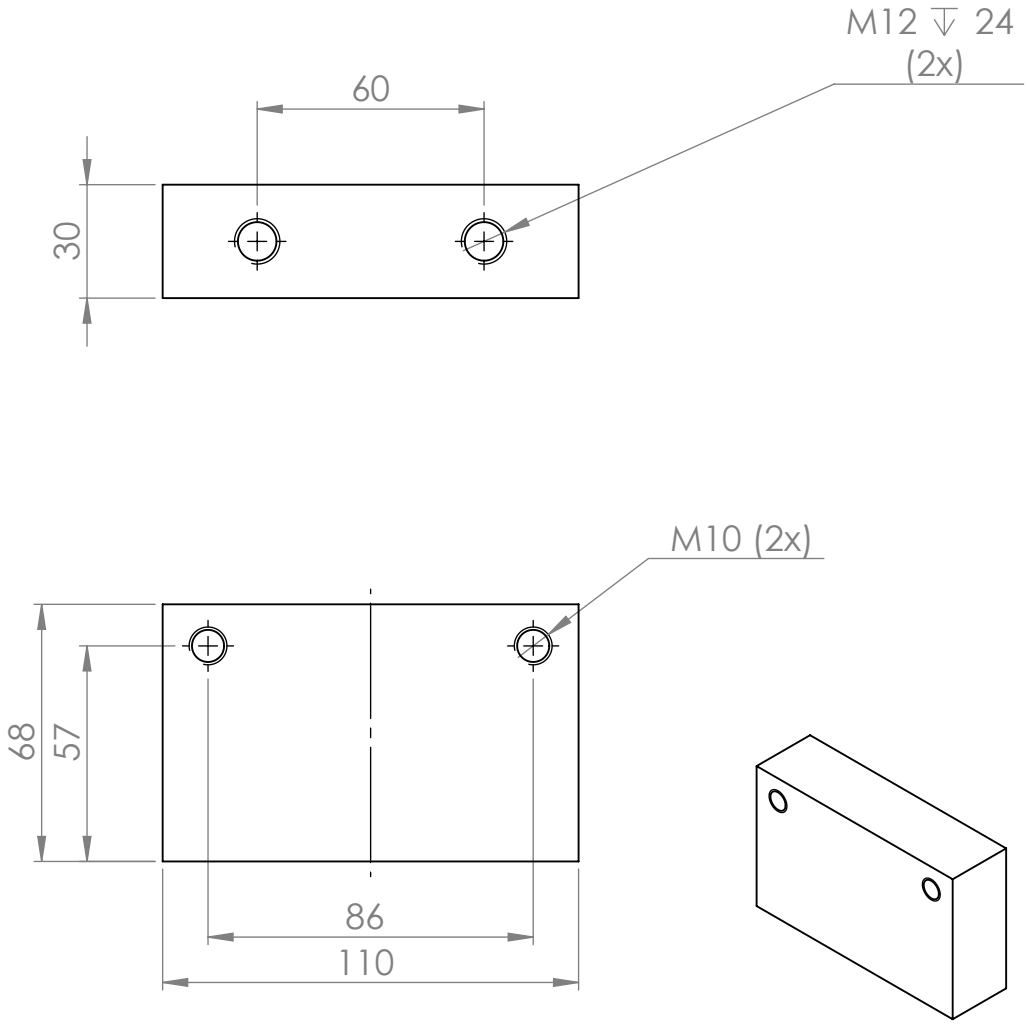




agujeros sin tolerancia ±0,2



	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	21.12.16			
REVISADO				
SK 43533-060			PLATO ALTURA VARIABLE	1:5
JOSE FRANCISCO CHUMAN ALVARADO				



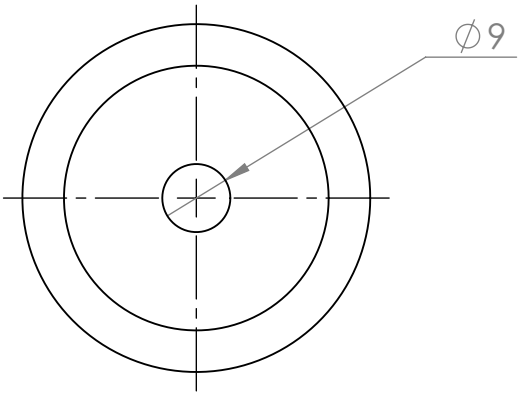
pavonado

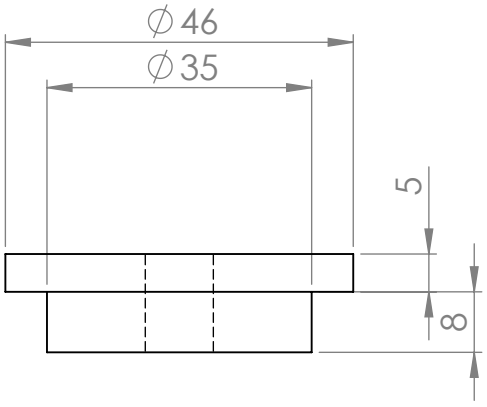
agujeros sin tolerancia ±0,2

2,2

-0,6

	FECHA	NOMBRE	
DIBUJADO	21.12.16		***UNIVERSIDAD DE PIURA***
REVISADO			*FACULTAD DE INGENIERIA*
SK 43533-061			1:2
JOSE FRANCISCO CHUMAN ALVARADO			SUJECIÓN FRENOS

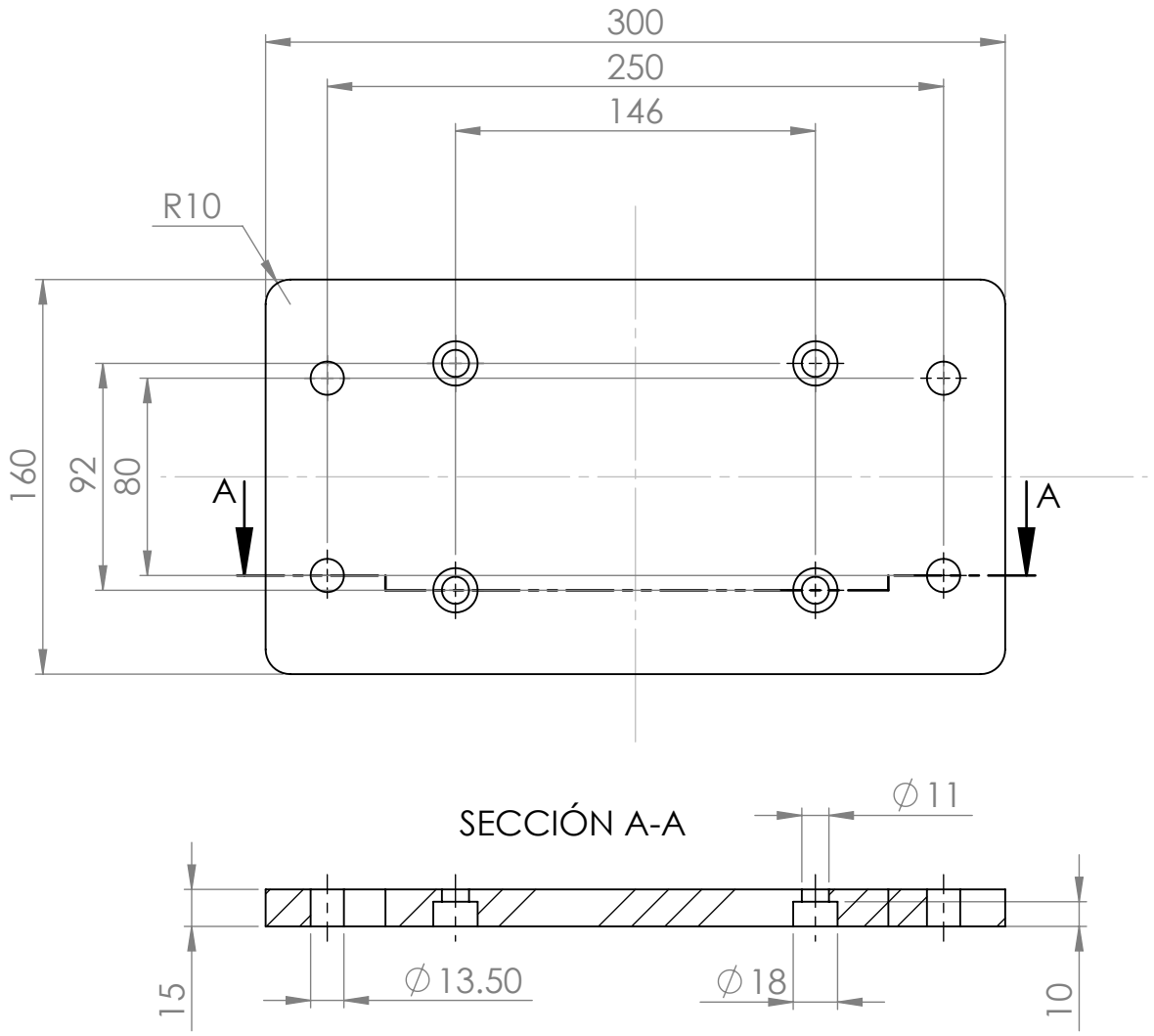




pavonados

3,2

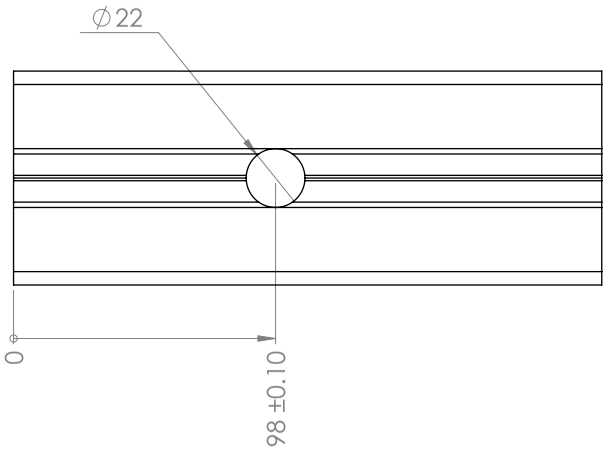
	FECHA	NOMBRE	
DIBUJADO	21.12.16		***UNIVERSIDAD DE PIURA***
REVISADO			*FACULTAD DE INGENIERIA*
SK 43533-062			TOPE 1:1
JOSE FRANCISCO CHUMAN ALVARADO			

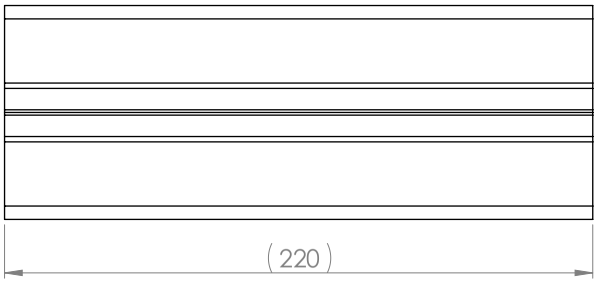


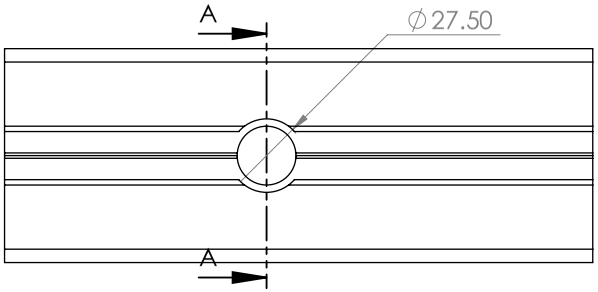
agujeros sin tolerancia ±0,2

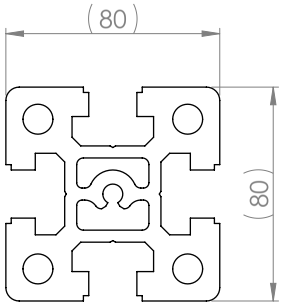
3.2 pavonado

	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	20.12.16			
REVISADO				
SK 43533-063			PLATO	1:3
JOSE FRANCISCO CHUMAN ALVARADO				

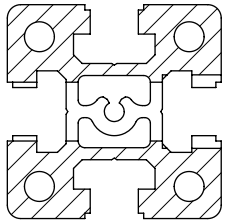






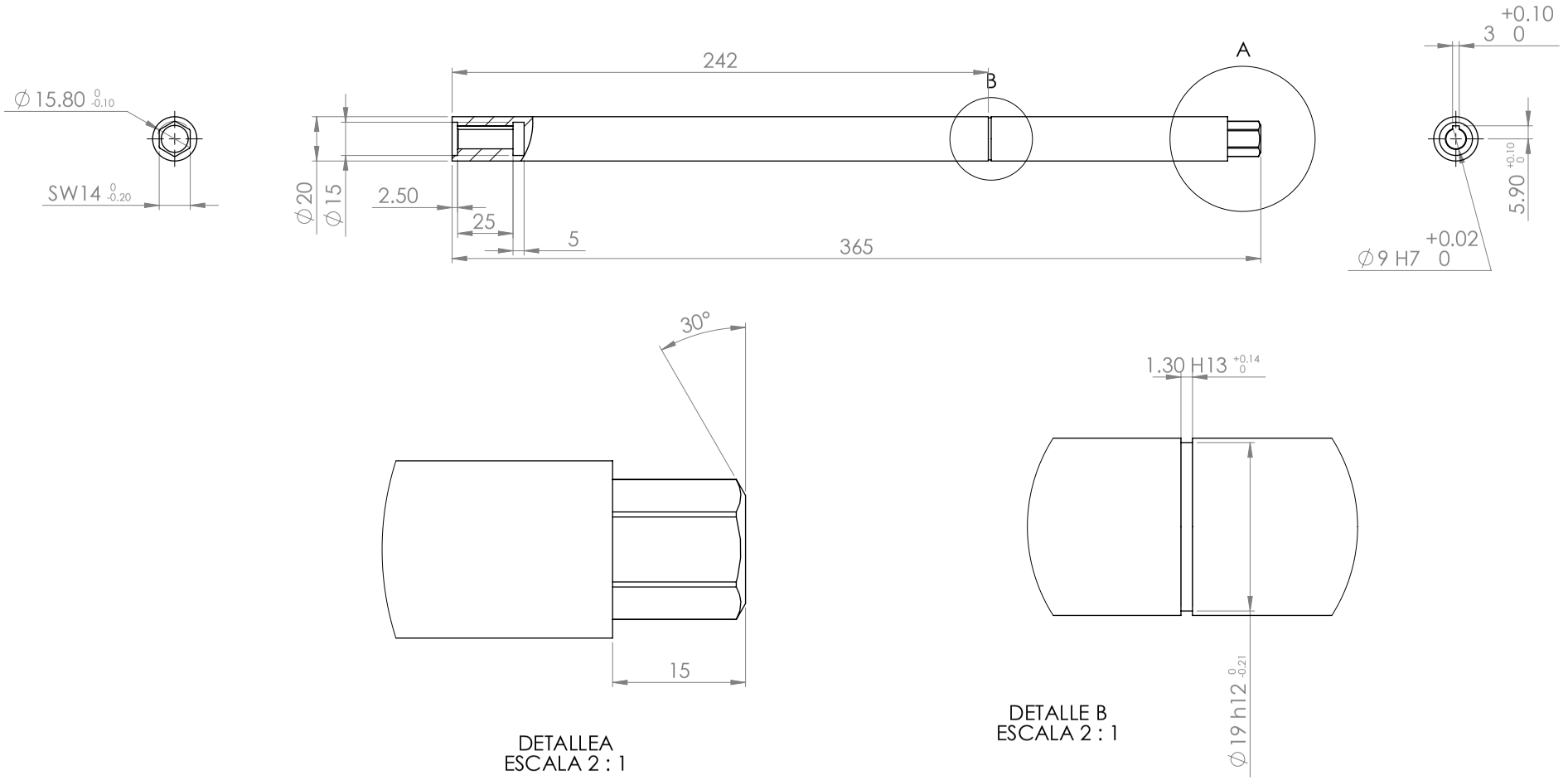


06



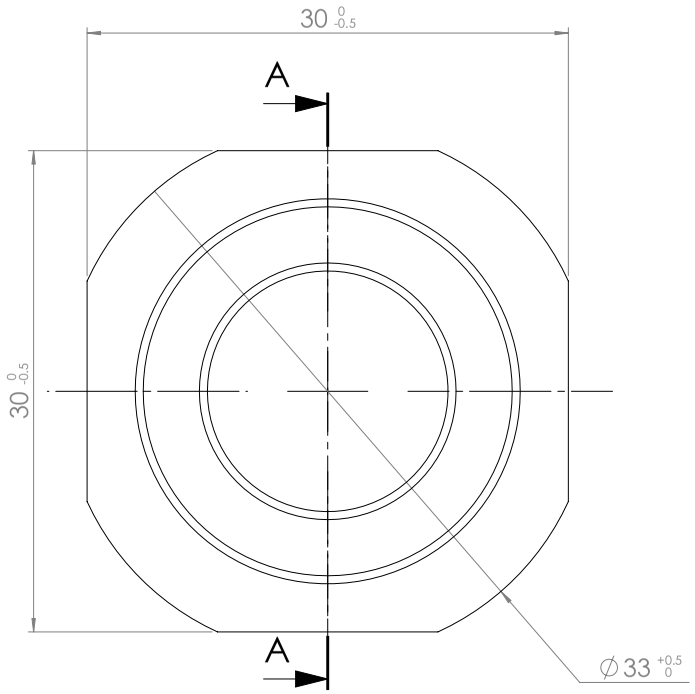
SECCIÓN A-A

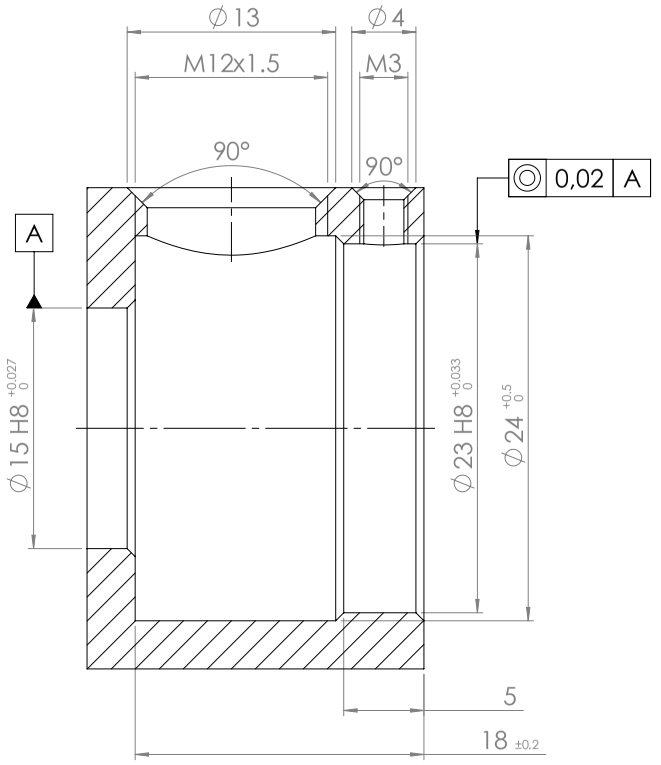
	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	21.12.16			
REVISADO				
SK 43533-064			PERFIL	1:2
JOSE FRANCISCO CHUMAN ALVARADO				



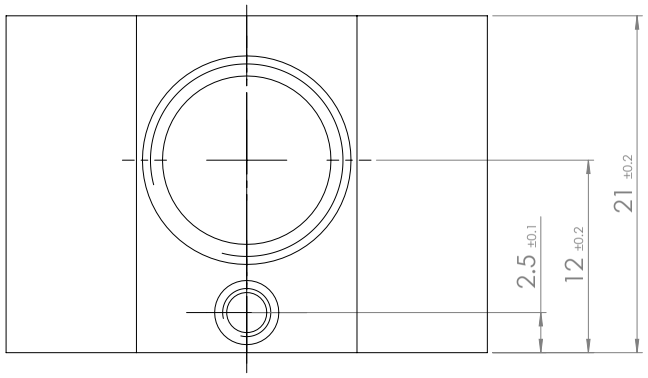
	FECHA	NOMBRE	
DIBUJADO	21.12.16		**UNIVERSIDAD DE PIURA**
REVISADO			*FACULTAD DE INGENIERIA*

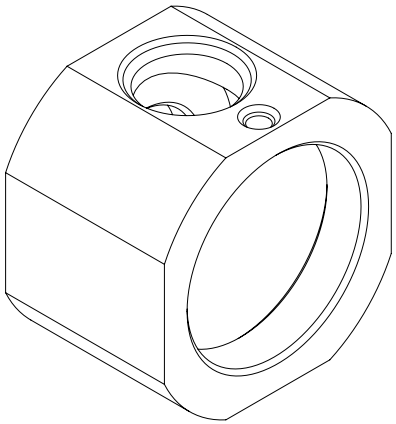
SK 43533-065	EXTENSIÓN	1:2
JOSE FRANCISCO CHUMAN ALVARADO		





SECCIÓNA-A

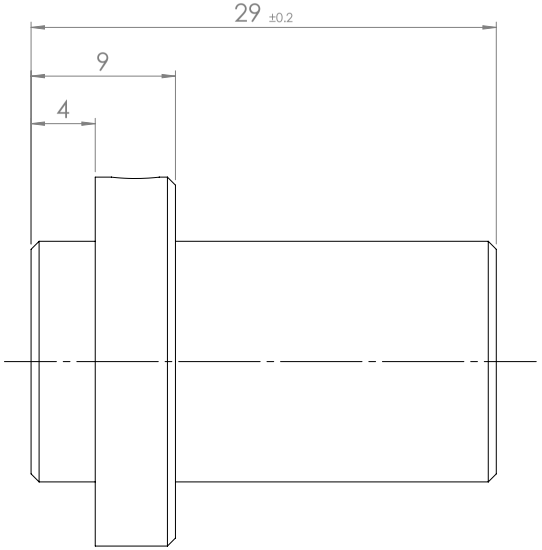


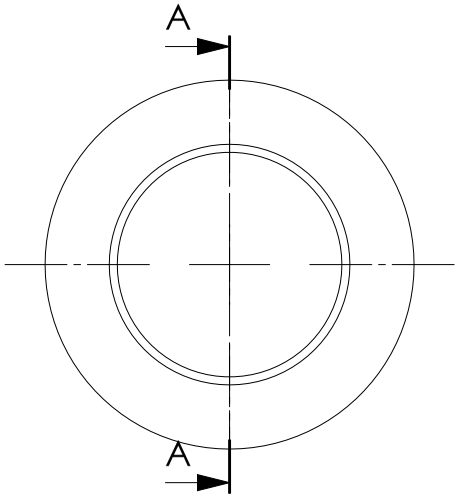


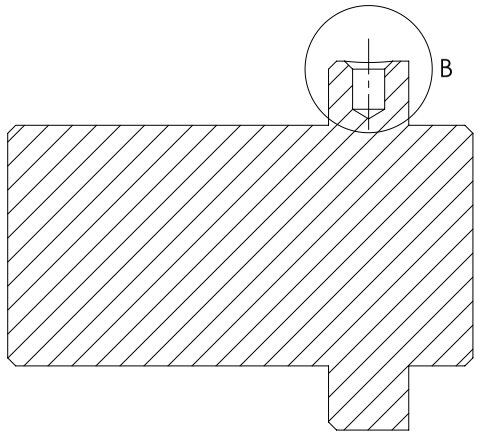
chaflanes sin medida 0.5x45°

N7

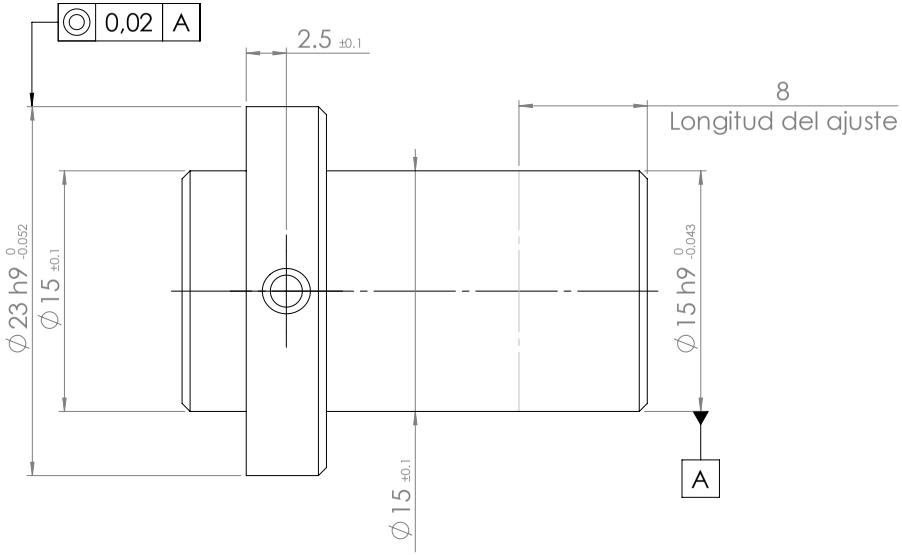
	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	10.01.17			
REVISADO				
SK 43533-066			CARCASA PEQUEÑA	3:1
JOSE FRANCISCO CHUMAN ALVARADO				

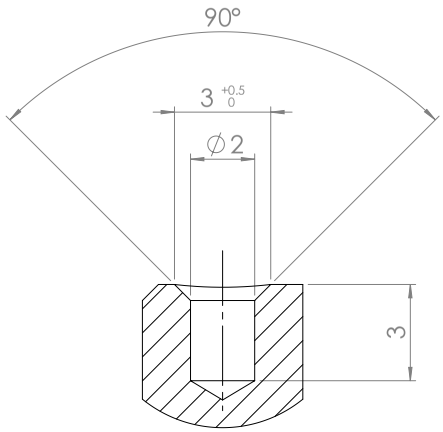






SECCIÓN A-A

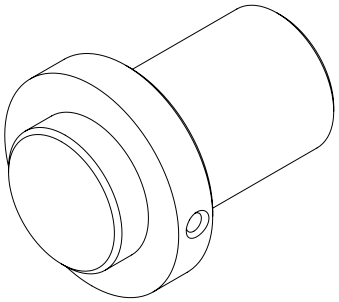




DETALLE B
ESCALA 6 : 1

chaflanes sin medida 0.5x45°

N7

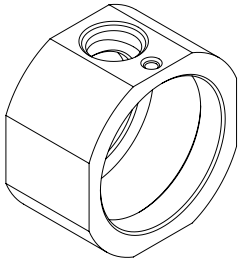


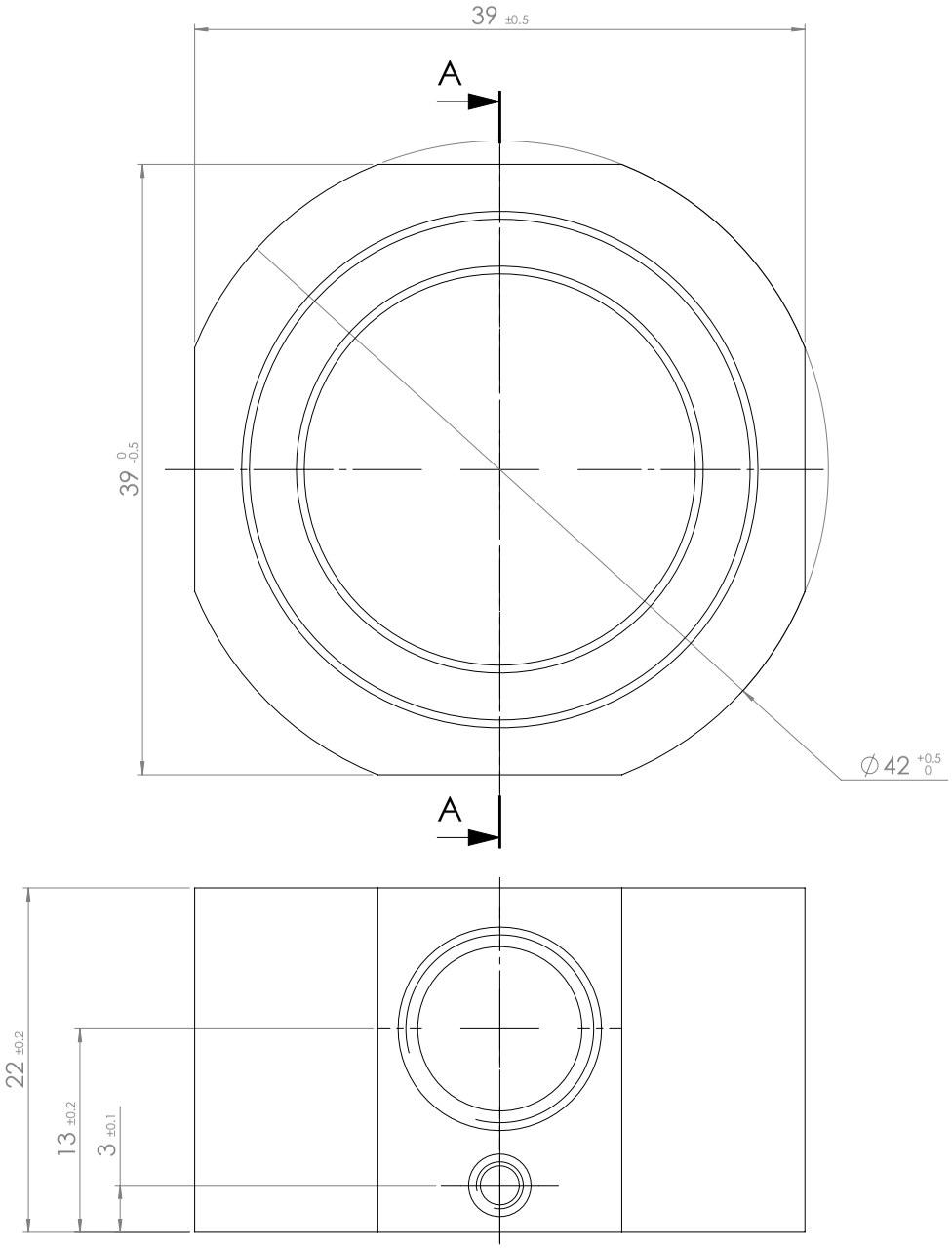
0,02 A

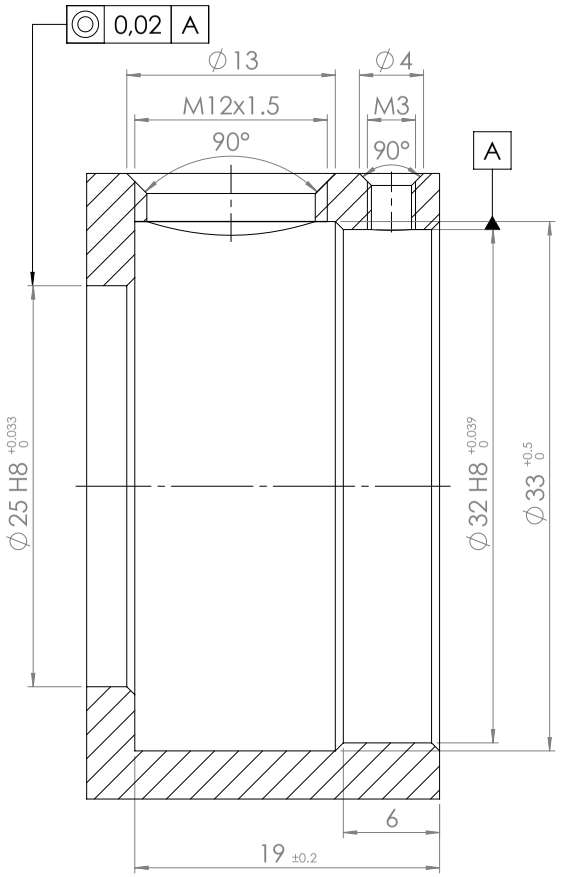
	FECHA	NOMBRE	
DIBUJADO	10.01.17		**UNIVERSIDAD DE PIURA**
REVISADO			*FACULTAD DE INGENIERIA*

SK 43533-067	SENSOR DE FUERZA PEQUEÑO	3:1
JOSE FRANCISCO CHUMAN ALVARADO		

93





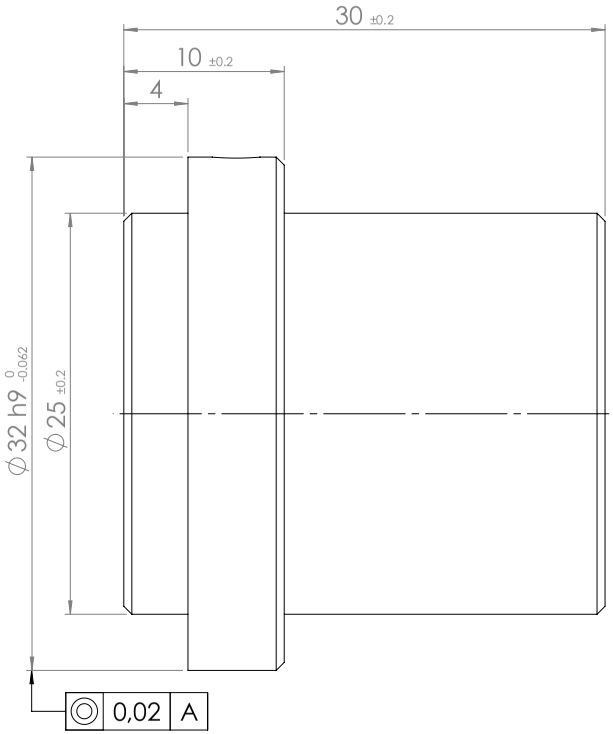


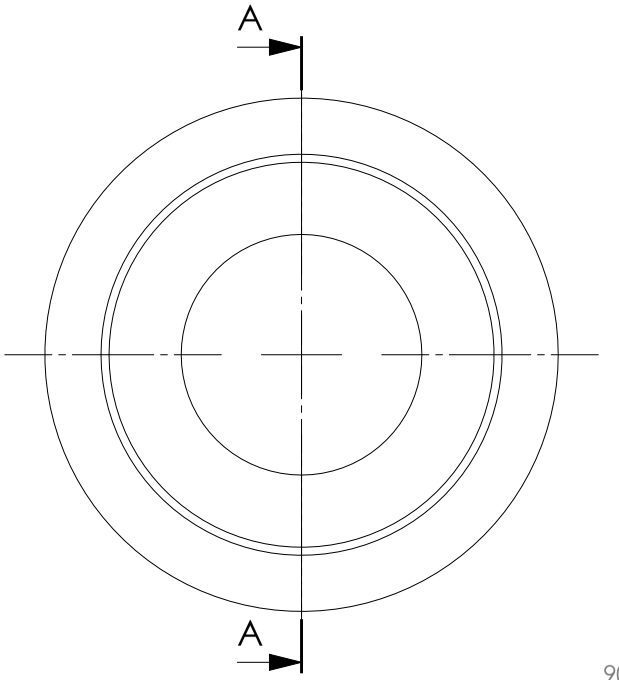
SECCIÓN A-A

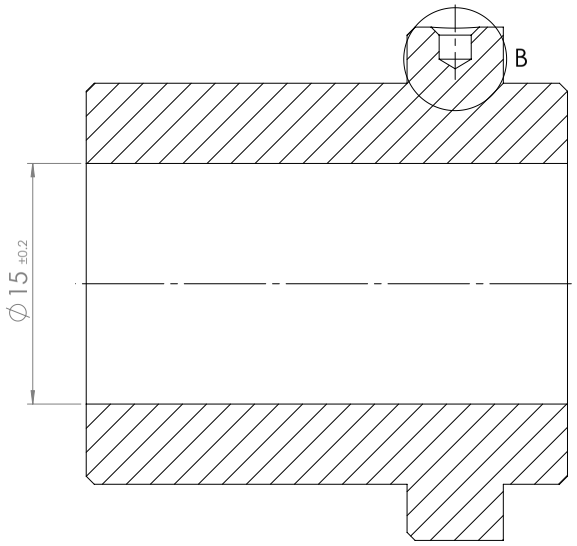
chaflanes sin medida 0.5x45°

N7

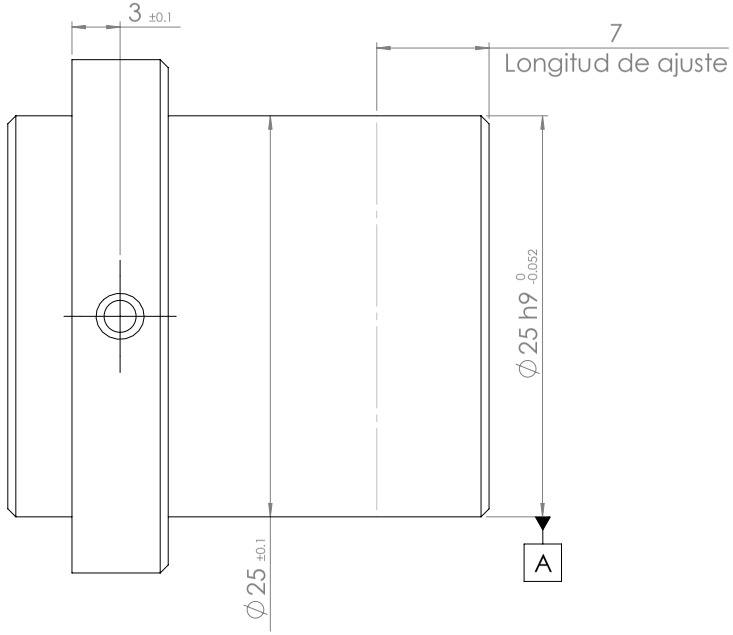
	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	10.01.17			
REVISADO				
SK 43533-068			CARCASA GRANDE	3:1
JOSE FRANCISCO CHUMAN ALVARADO				

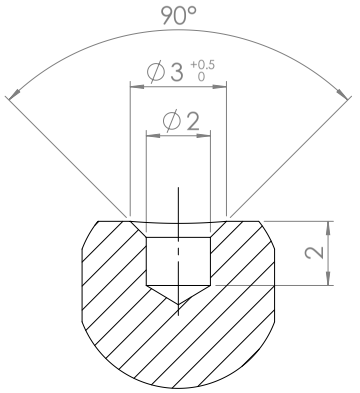






SECCIÓN A-A





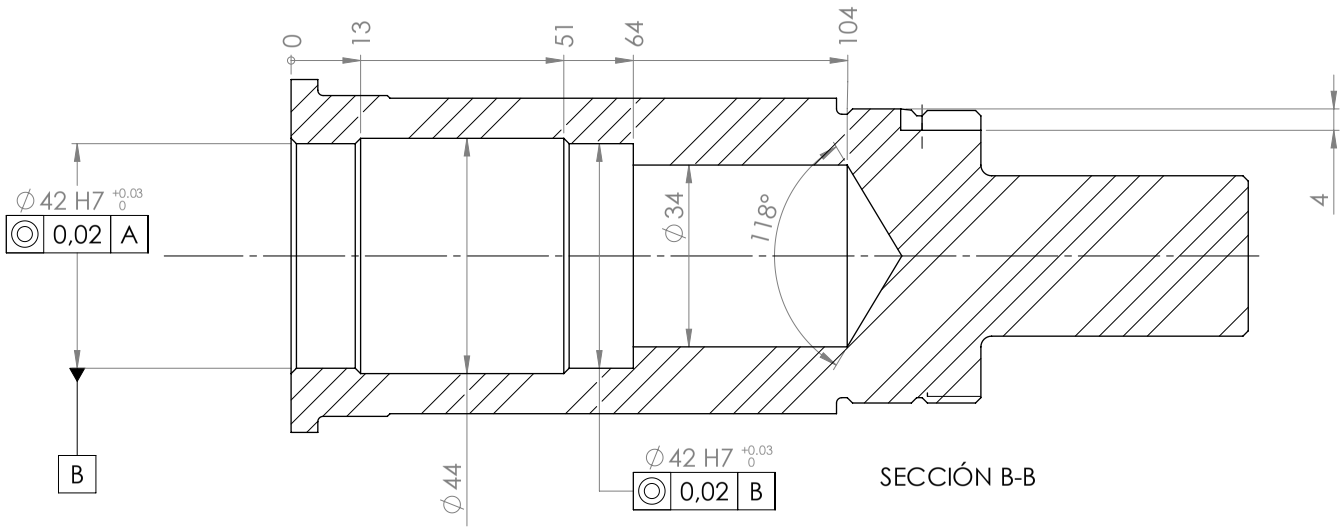
DETALLE B
ESCALA 6 : 1

chaflanes sin medida 0.5x45°

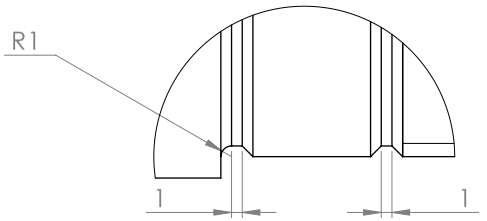
N7

	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	10.01.17			
REVISADO				
SK 43533-069			SENSOR DE FUERZA GRANDE	3:1
JOSE FRANCISCO CHUMAN ALVARADO				

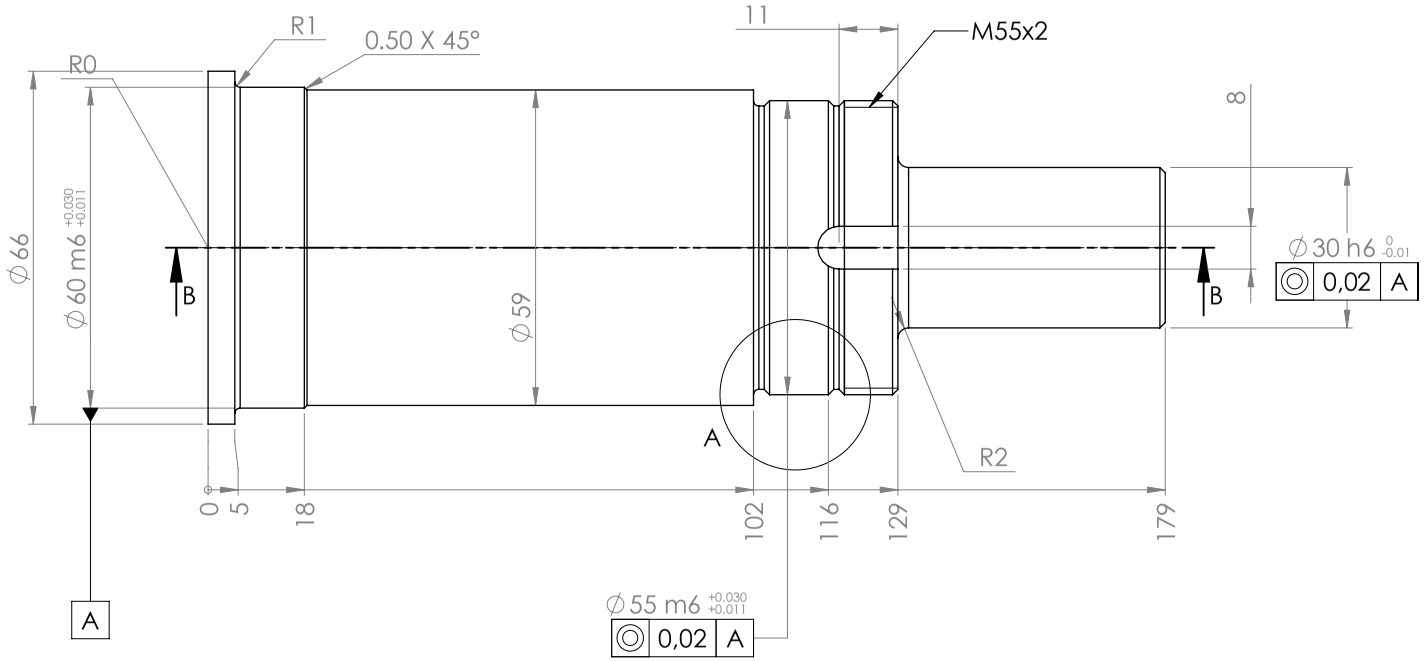
95

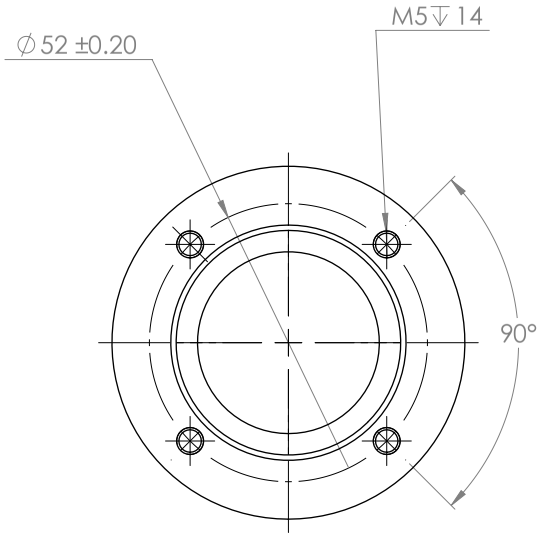


SECCIÓN B-B



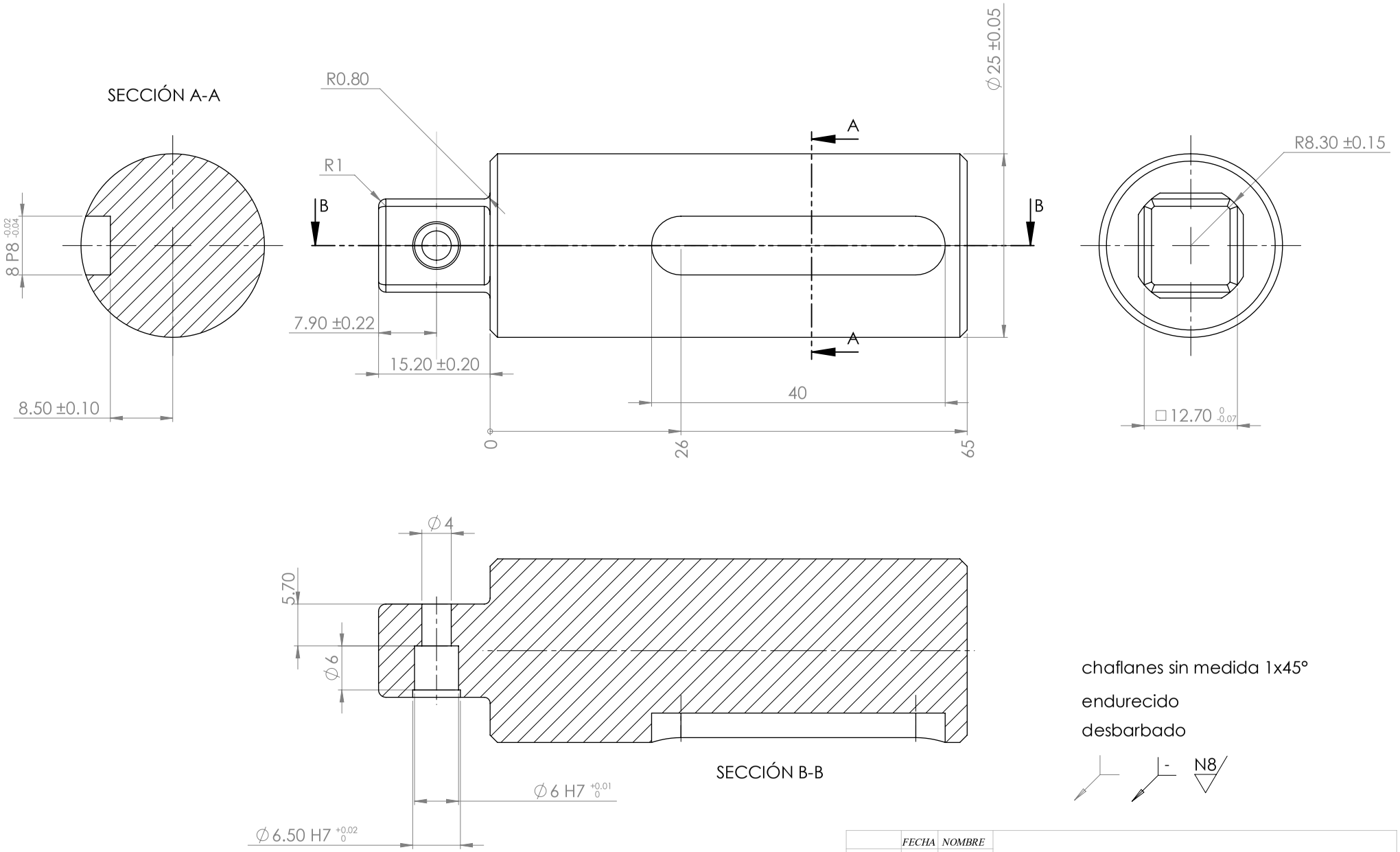
DETALLE A
ESCALA 2 : 1





chaflanes sin medida 1x45°
agujeros sin tolerancia ±0,2

	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	22.12.16			
REVISADO				
sk43535-17			CARCASA	1:1
JOSE FRANCISCO CHUMAN ALVARADO				



chaflanes sin medida 1x45°

endurecido

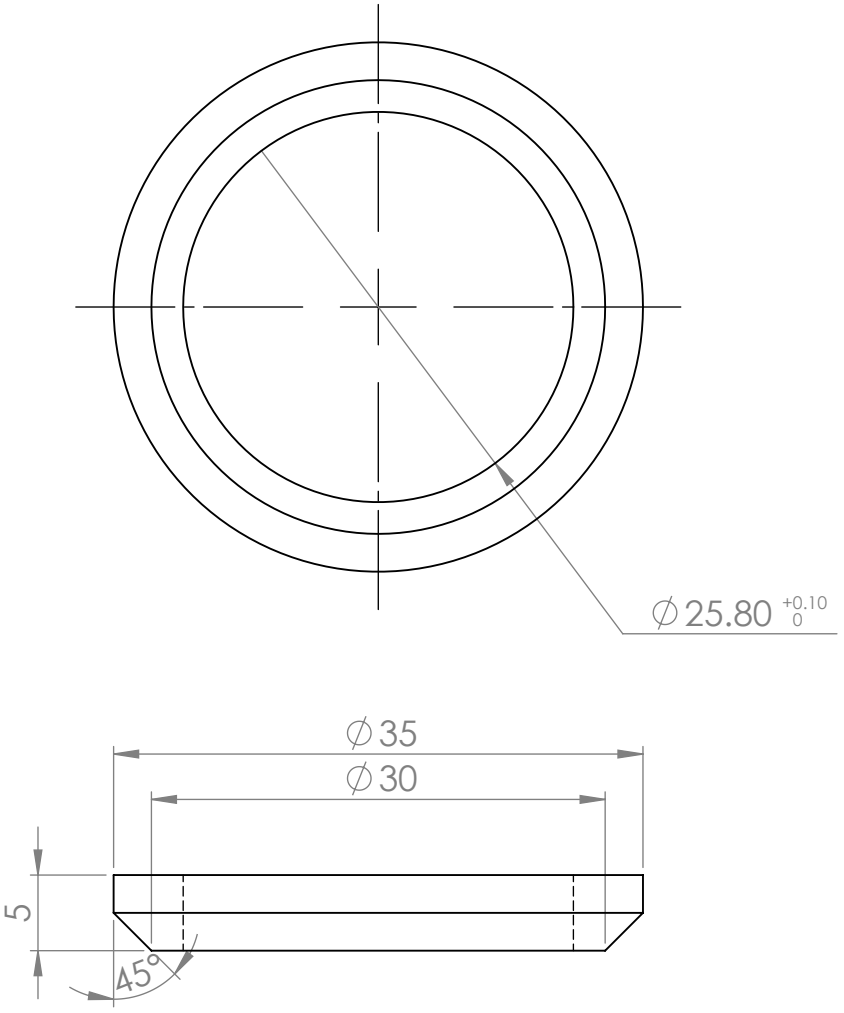
desbarbado

N8

	FECHA	NOMBRE	UNIVERSIDAD DE PIURA
DIBUJADO	23.12.16		FACULTAD DE INGENIERIA
REVISADO			

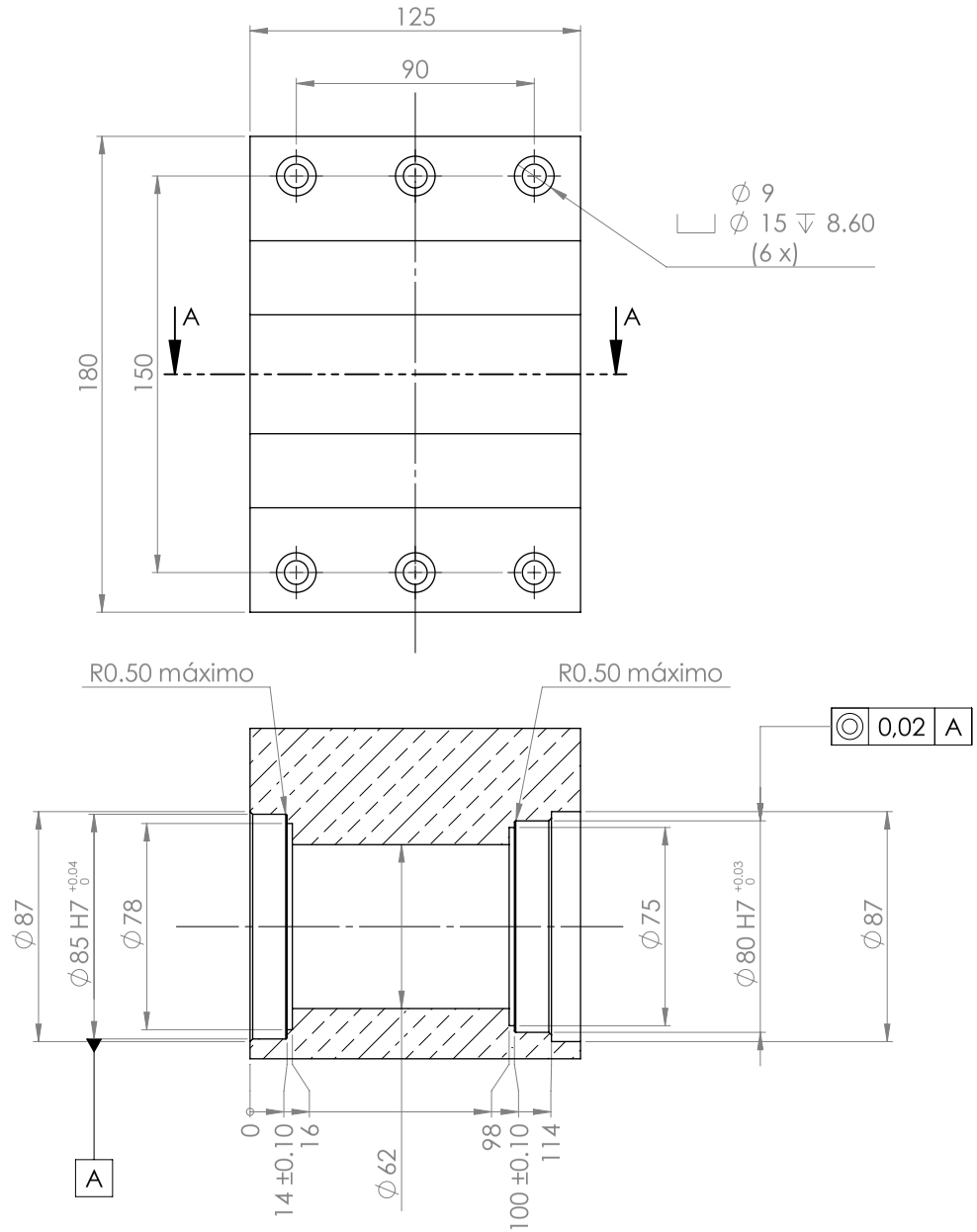
sk43535-19	MACHO	2:1
JOSE FRANCISCO CHUMAN ALVARADO		

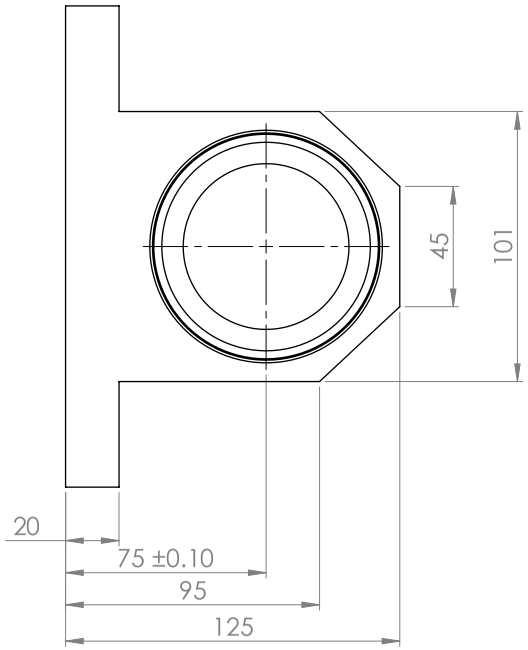
97



pavonado

	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	23.12.16			
REVISADO				
sk43535-20			TOPE RESORTE EJE	2:1
JOSE FRANCISCO CHUMAN ALVARADO				

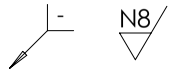




arenado

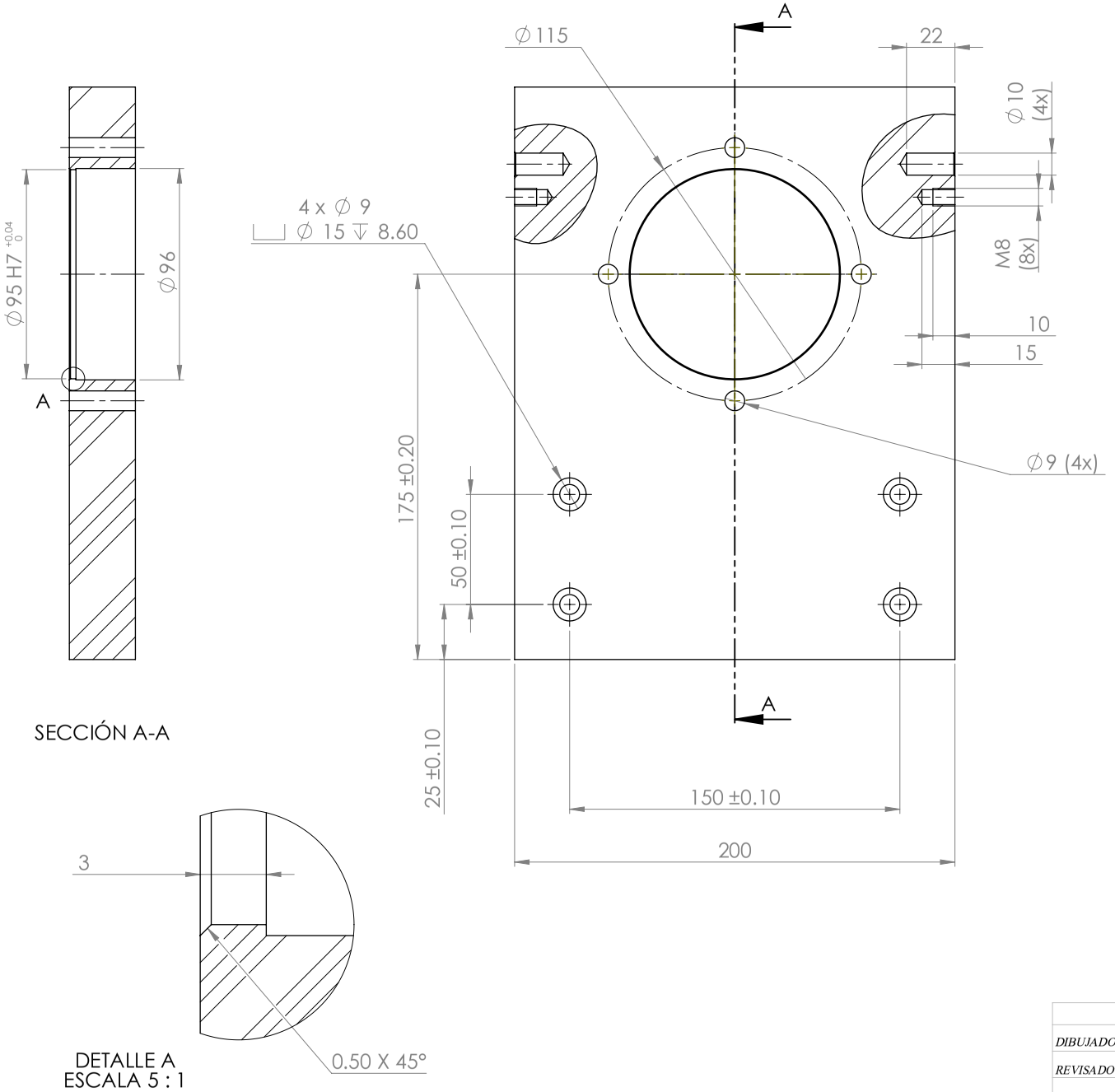
agujeros sin tolerancia ±0,2

chaflanes sin medida 1x45°



	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	23.12.16			
REVISADO				
sk43535-21			SUJETOR	1:2
JOSE FRANCISCO CHUMAN ALVARADO				

99



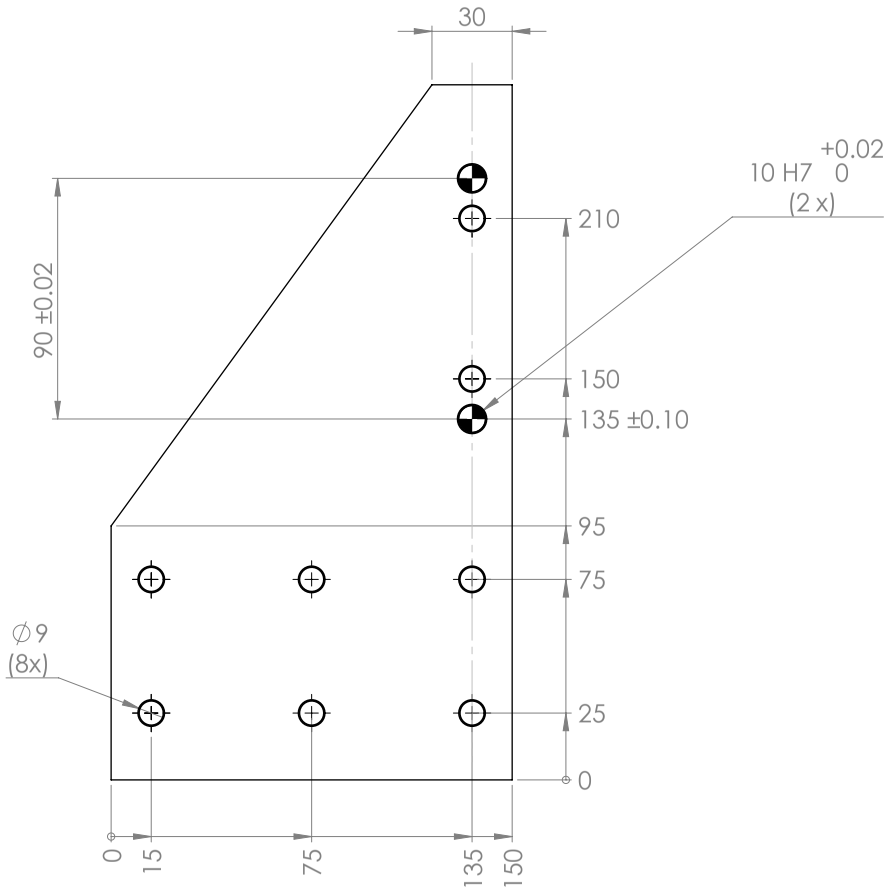
100

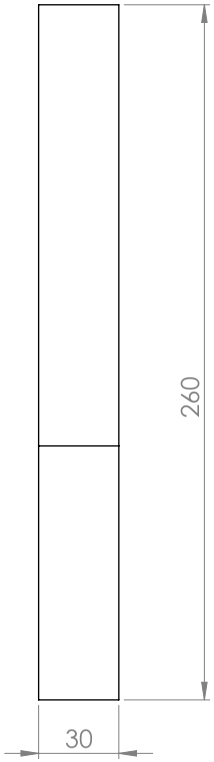
pavonado

agujeros sin tolerancia ±0,2

N8

	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	23.12.16			
REVISADO				
sk43535-22			PLATO PRINCIPAL	1:2
JOSE FRANCISCO CHUMAN ALVARADO				

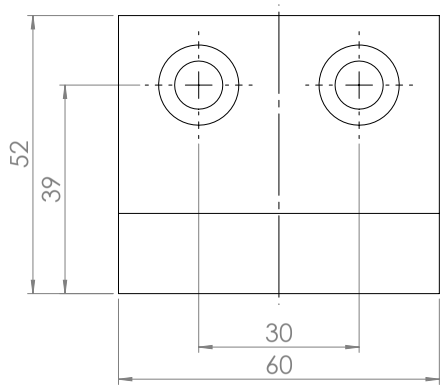


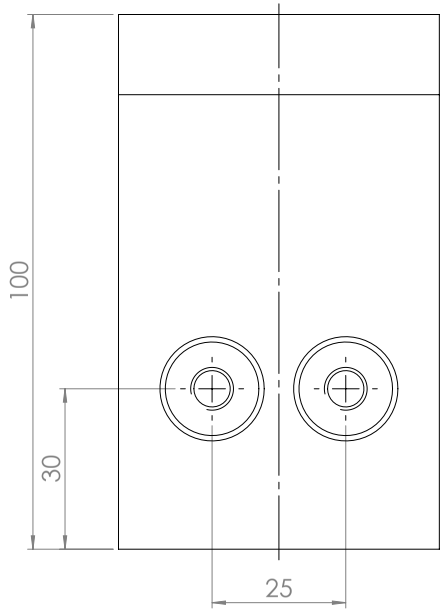


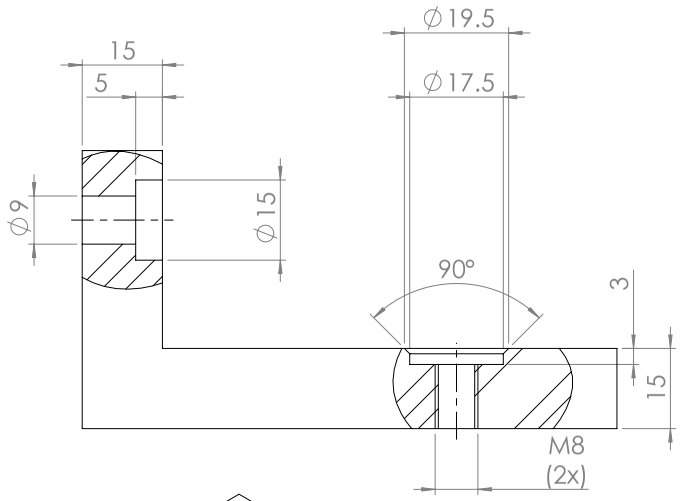
pavonado
agujeros sin tolerancia ±0,2

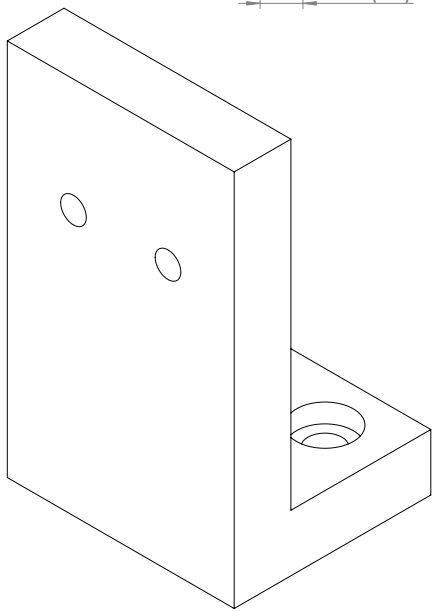
N8

	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	22.12.16			101
REVISADO				
sk43535-23			PLATO LATERAL	1:2
JOSE FRANCISCO CHUMAN ALVARADO				







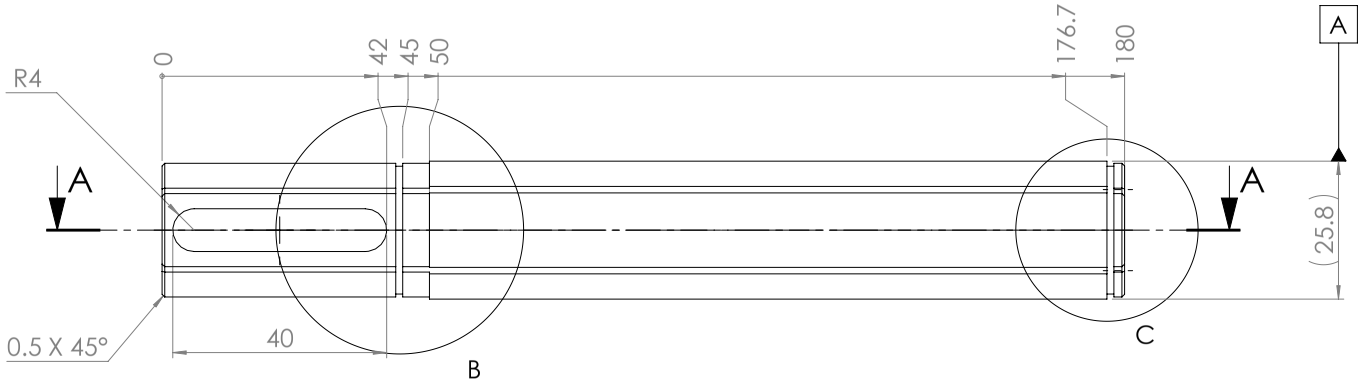


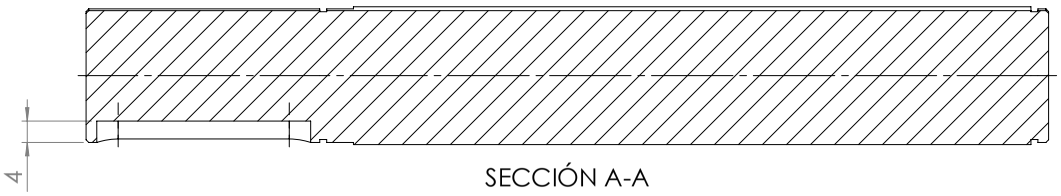
arenado

pavonado

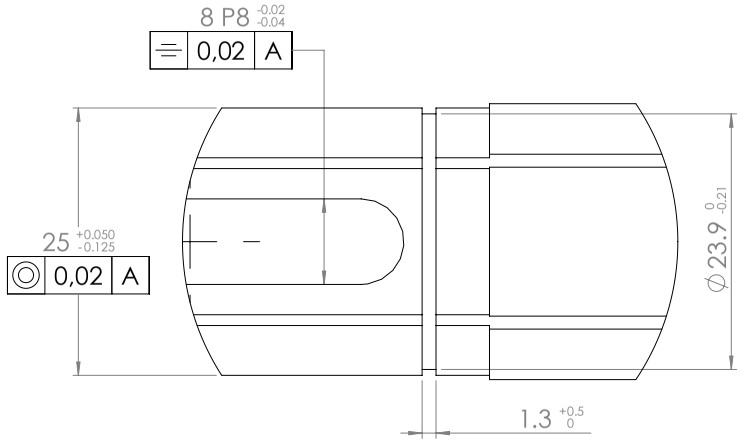
agujeros sin tolerancia ±0,2

	FECHA	NOMBRE	UNIVERSIDAD DE PIURA FACULTAD DE INGENIERIA	
DIBUJADO	09.01.17			
REVISADO				
sk43535-24			CONEXIÓN FLEXIBLE	1:1
JOSE FRANCISCO CHUMAN ALVARADO				

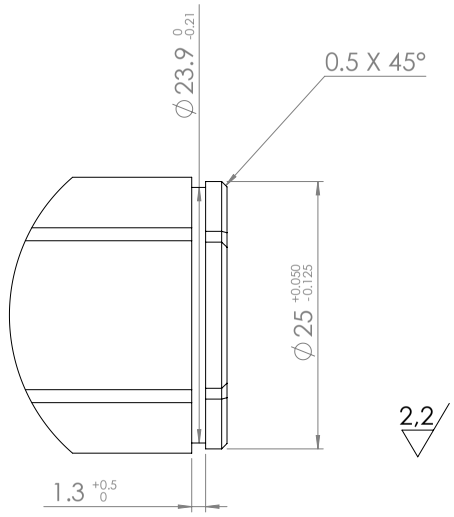




SECCIÓN A-A



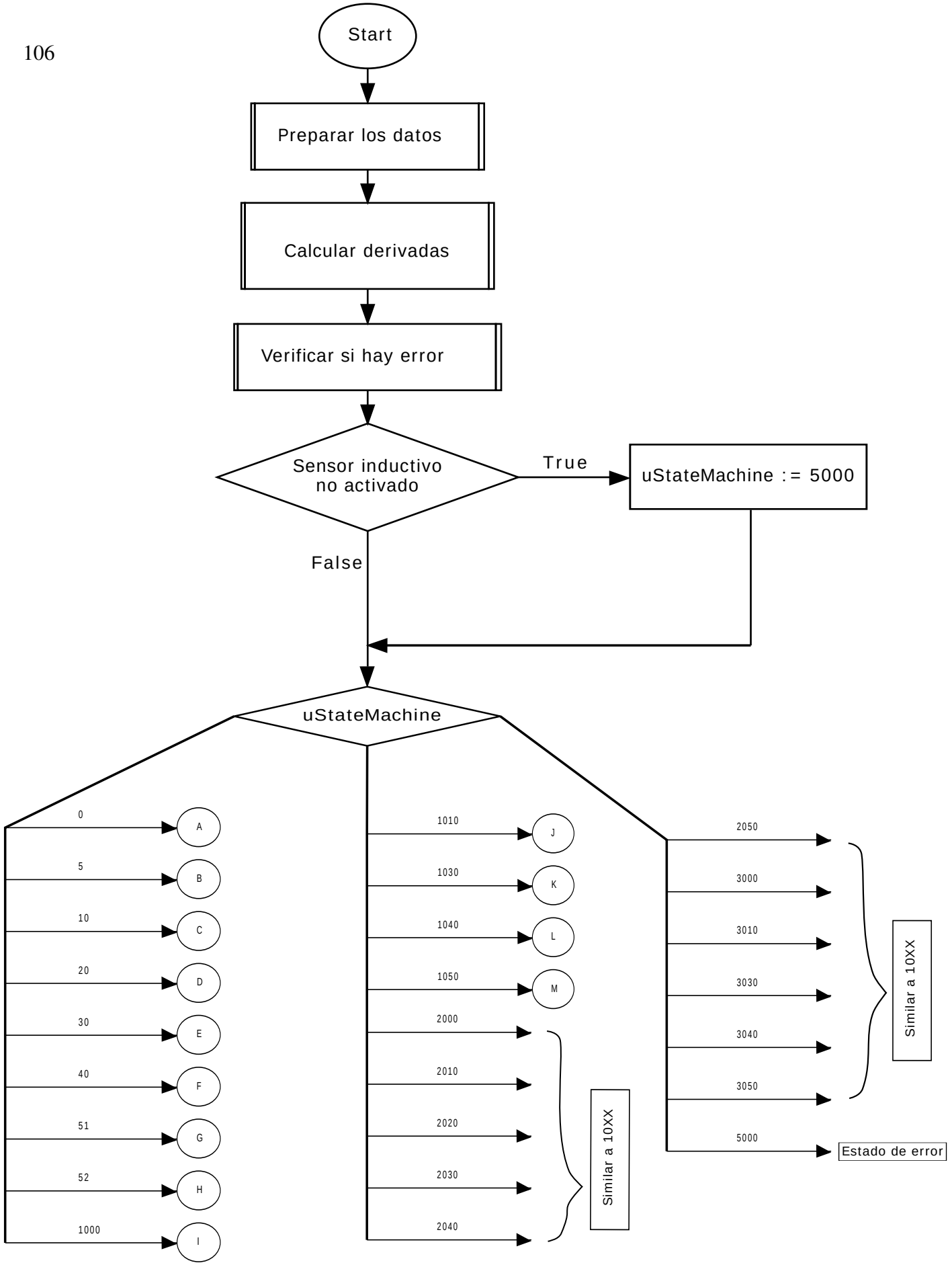
DETALLE B
ESCALA 2 : 1

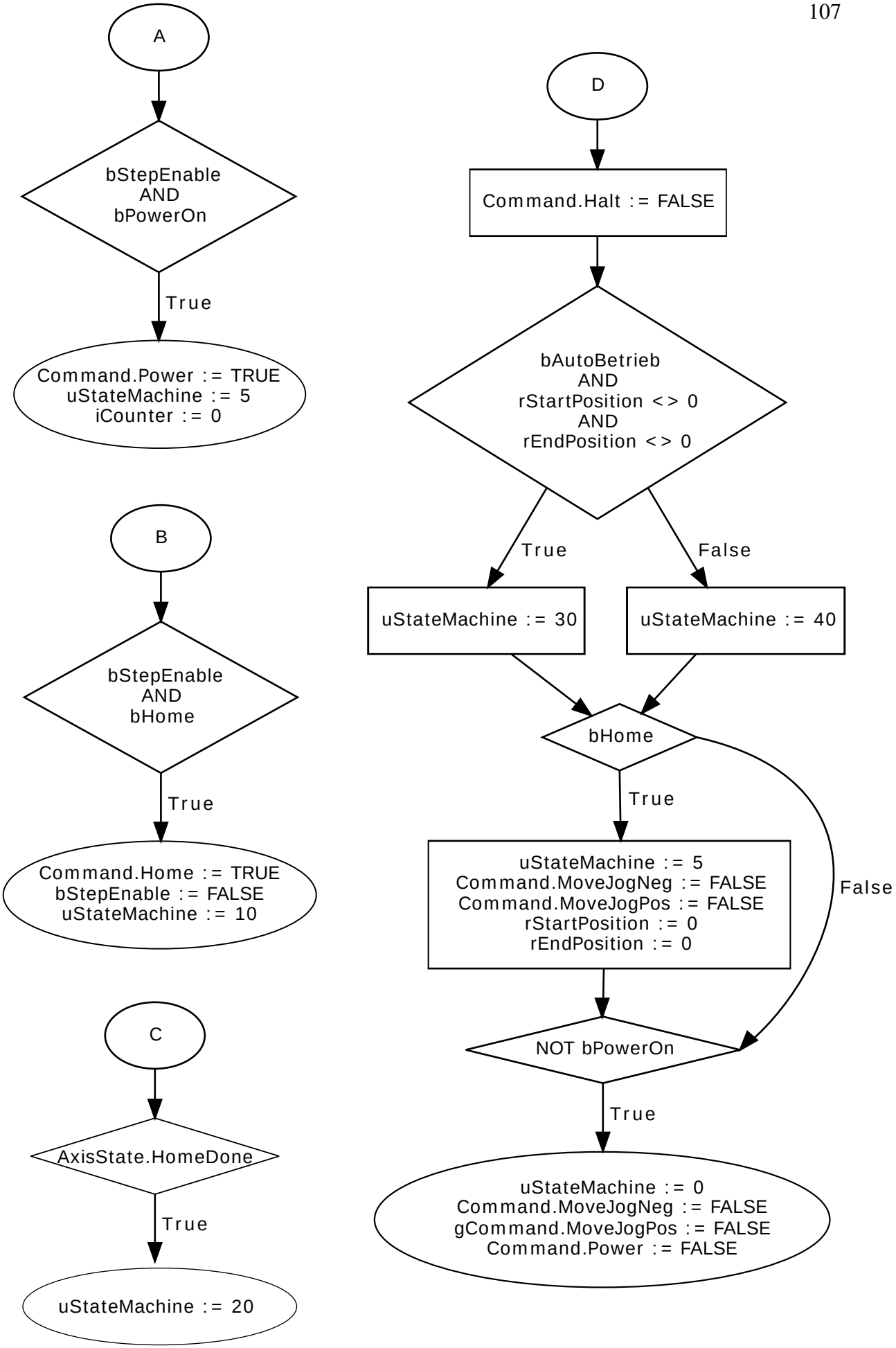


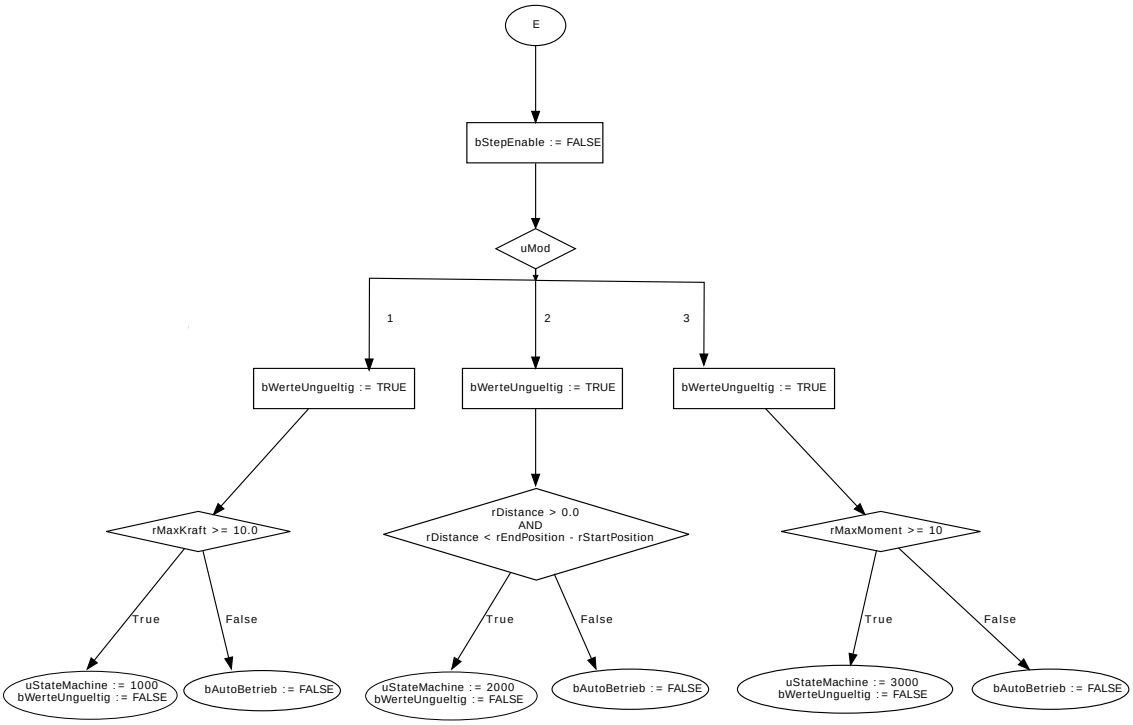
DETALLE C
ESCALA 2 : 1

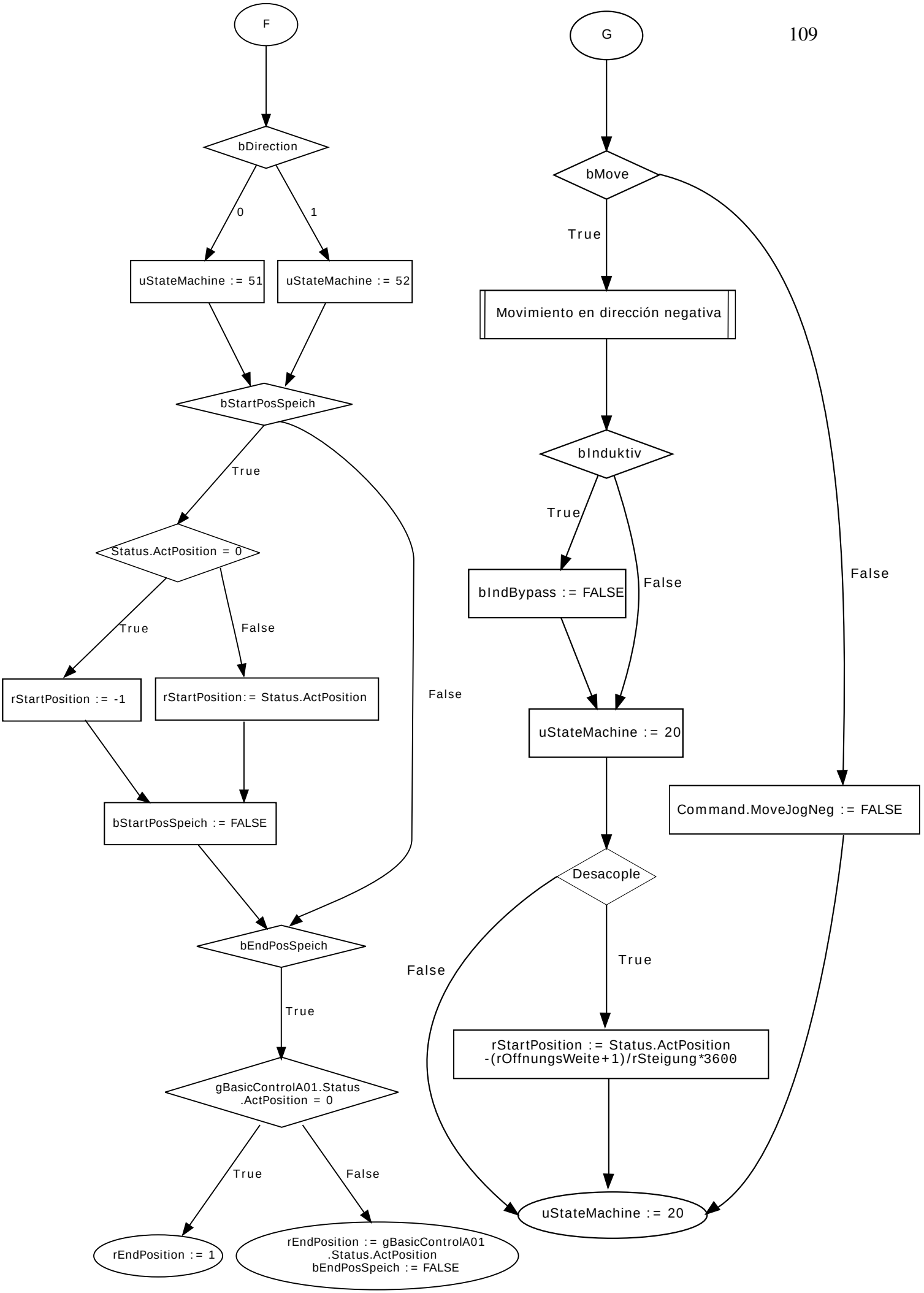
	FECHA	NOMBRE	UNIVERSIDAD DE PIURA	
DIBUJADO	16.01.17		FACULTAD DE INGENIERIA	103
REVISADO				
sk43535-25			EJE TORQUE	1:1
JOSE FRANCISCO CHUMAN ALVARADO				

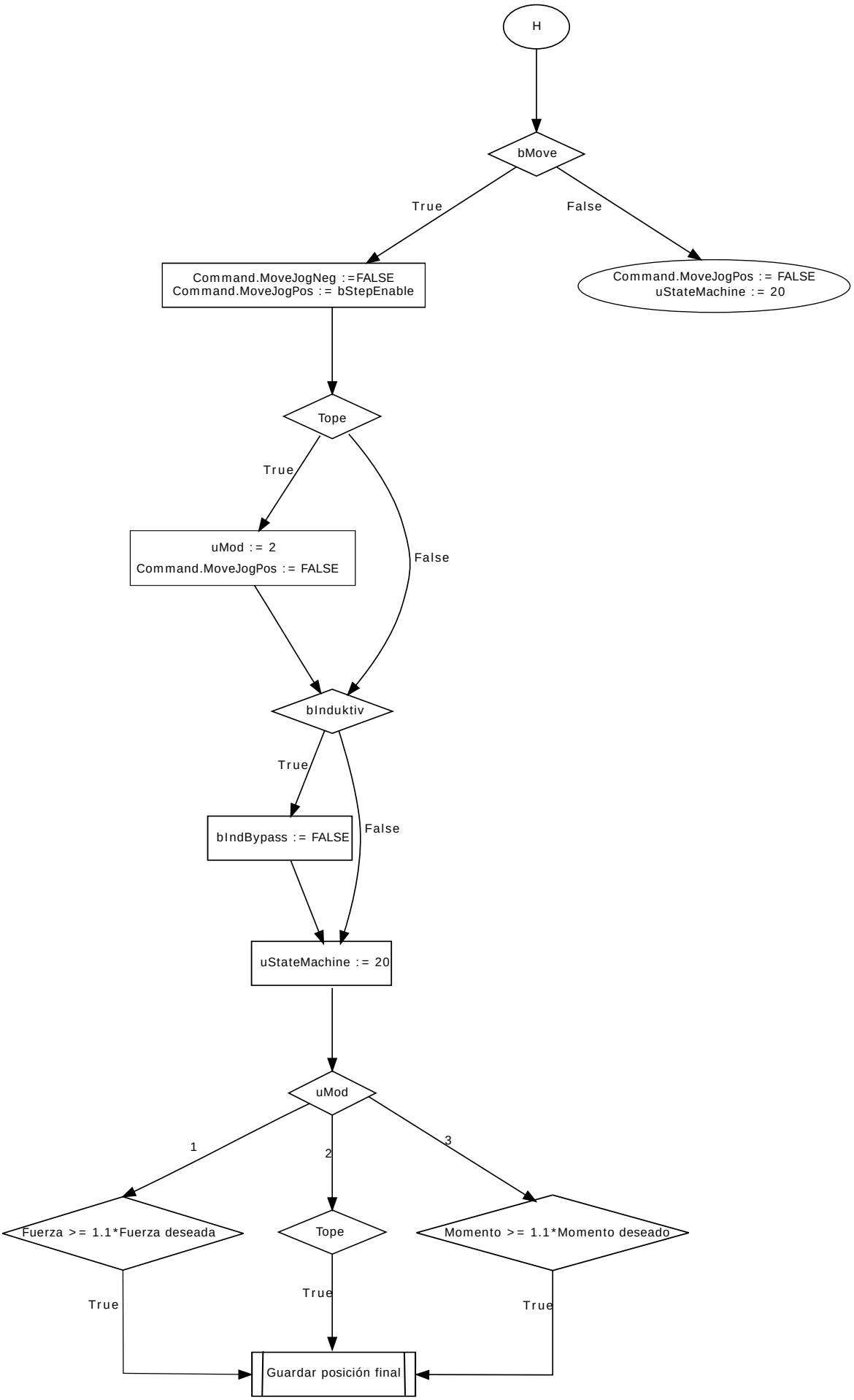
Apéndice B
Diagrama de flujo

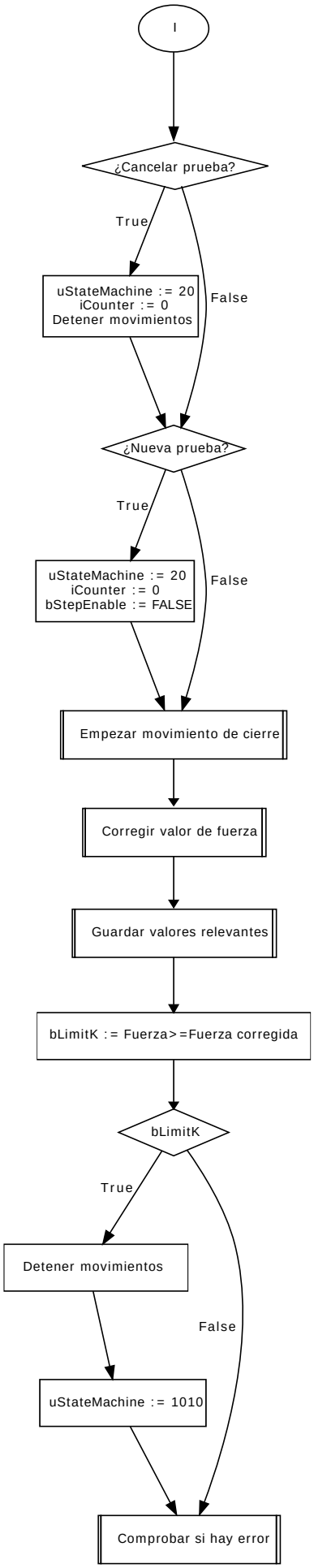

Start
Preparar los datos
Calcular derivadas
Verificar si hay error
Sensor inductivo no activado
True
uStateMachine := 5000
False
uStateMachine
0
A
5
B
10
C
20
D
30
E
40
F
51
G
52
H
1000
I
1010
J
1030
K
1040
L
1050
M
2000
2010
2020
2030
2040
Similar a 10XX
2050
3000
3010
3030
3040
3050
Similar a 10XX
5000
Estado de error

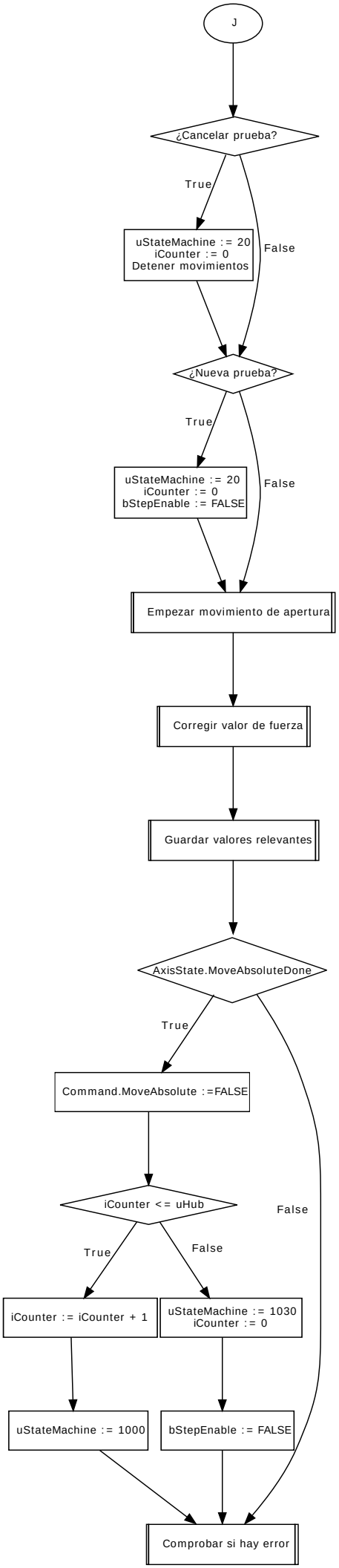

A
bStepEnable
AND
bPowerOn
True
Command.Power := TRUE
uStateMachine := 5
iCounter := 0
B
bStepEnable
AND
bHome
True
Command.Home := TRUE
bStepEnable := FALSE
uStateMachine := 10
C
AxisState.HomeDone
True
uStateMachine := 20
D
Command.Halt := FALSE
bAutoBetrieb
AND
rStartPosition <> 0
AND
rEndPosition <> 0
True
False
uStateMachine := 30
uStateMachine := 40
bHome
True
False
uStateMachine := 5
Command.MoveJogNeg := FALSE
Command.MoveJogPos := FALSE
rStartPosition := 0
rEndPosition := 0
NOT bPowerOn
True
uStateMachine := 0
Command.MoveJogNeg := FALSE
gCommand.MoveJogPos := FALSE
Command.Power := FALSE

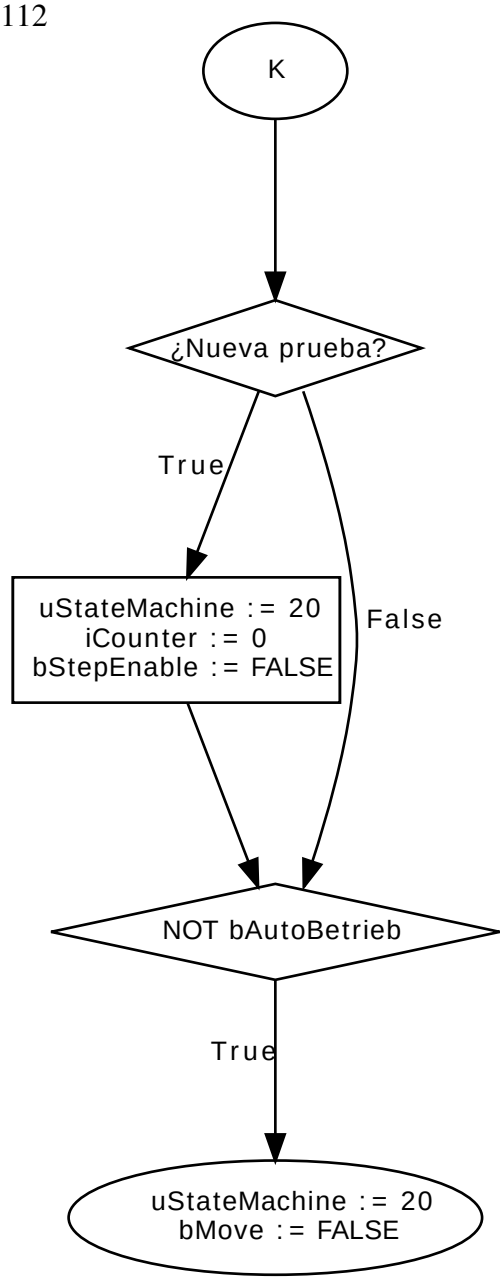

E
bStepEnable := FALSE
uMod
1
2
3
bWerteUngueltig := TRUE
bWerteUngueltig := TRUE
bWerteUngueltig := TRUE
rMaxKraft >= 10.0
rDistance > 0.0
AND
rDistance < rEndPosition - rStartPosition
rMaxMoment >= 10
True
False
True
False
True
False
uStateMachine := 1000
bWerteUngueltig := FALSE
bAutoBetrieb := FALSE
uStateMachine := 2000
bWerteUngueltig := FALSE
bAutoBetrieb := FALSE
uStateMachine := 3000
bWerteUngueltig := FALSE
bAutoBetrieb := FALSE

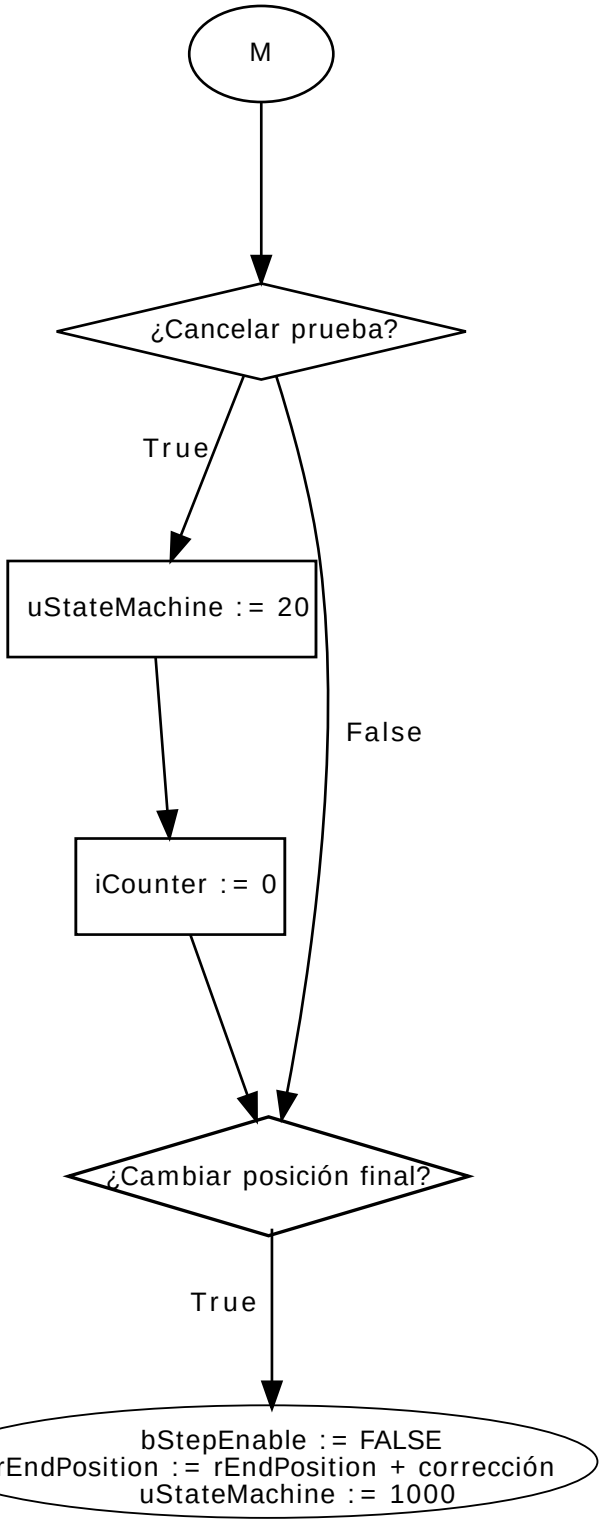

F
bDirection
0
1
uStateMachine := 51
uStateMachine := 52
bStartPosSpeich
True
Status.ActPosition = 0
True
False
rStartPosition := -1
rStartPosition:= Status.ActPosition
bStartPosSpeich := FALSE
False
bEndPosSpeich
True
gBasicControlA01.Status
.ActPosition = 0
True
False
rEndPosition := 1
rEndPosition := gBasicControlA01
.Status.ActPosition
bEndPosSpeich := FALSE
G
bMove
True
Movimiento en dirección negativa
bInduktiv
True
False
bIndBypass := FALSE
uStateMachine := 20
False
Command.MoveJogNeg := FALSE
Desacople
True
False
rStartPosition := Status.ActPosition
-(rOffnungsWeite+1)/rSteigung*3600
uStateMachine := 20

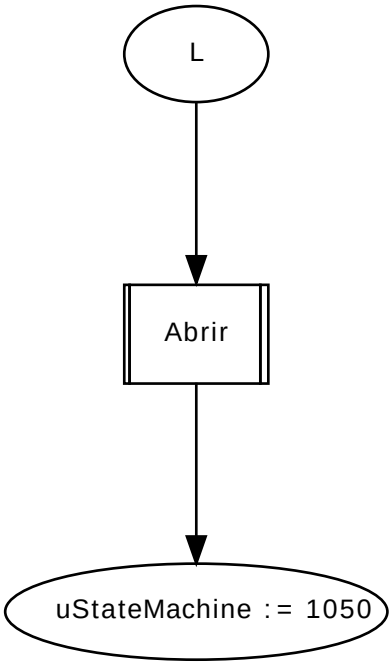

H
bMove
True
False
Command.MoveJogNeg :=FALSE
Command.MoveJogPos := bStepEnable
Command.MoveJogPos := FALSE
uStateMachine := 20
Tope
True
False
uMod := 2
Command.MoveJogPos := FALSE
bInduktiv
True
False
bIndBypass := FALSE
uStateMachine := 20
uMod
1
2
3
Fuerza >= 1.1*Fuerza deseada
Tope
Momento >= 1.1*Momento deseado
True
True
True
Guardar posición final


I
¿Cancelar prueba?
True
False
uStateMachine := 20
iCounter := 0
Detener movimientos
¿Nueva prueba?
True
False
uStateMachine := 20
iCounter := 0
bStepEnable := FALSE
Empezar movimiento de cierre
Corregir valor de fuerza
Guardar valores relevantes
bLimitK := Fuerza>=Fuerza corregida
bLimitK
True
False
Detener movimientos
uStateMachine := 1010
Comprobar si hay error


J
¿Cancelar prueba?
True
False
uStateMachine := 20
iCounter := 0
Detener movimientos
¿Nueva prueba?
True
False
uStateMachine := 20
iCounter := 0
bStepEnable := FALSE
Empezar movimiento de apertura
Corregir valor de fuerza
Guardar valores relevantes
AxisState.MoveAbsoluteDone
True
False
Command.MoveAbsolute :=FALSE
iCounter <= uHub
True
False
iCounter := iCounter + 1
uStateMachine := 1030
iCounter := 0
uStateMachine := 1000
bStepEnable := FALSE
Comprobar si hay error


K
¿Nueva prueba?
True
False
uStateMachine := 20
iCounter := 0
bStepEnable := FALSE
NOT bAutoBetrieb
True
uStateMachine := 20
bMove := FALSE


M
¿Cancelar prueba?
True
False
uStateMachine := 20
iCounter := 0
¿Cambiar posición final?
True
bStepEnable := FALSE
rEndPosition := rEndPosition + corrección
uStateMachine := 1000


L
Abrir
uStateMachine := 1050

Apéndice C
Lista de variables

Tabla 9: Variablen Liste

Nombre	Tipo	Descripción
abKV	BOOL[0..16]	
arMKraft	REAL[0..16]	
arMMoment	REAL[0..16]	Lista de
arSteigung	REAL[0..16]	parámetros
asSpanner	STRING[80][0..16]	predefinidos
sSpanner	STRING[80]	
uAuswahl	USINT	Cuál de las opciones de la lista fue escogida
rGMoment	REAL	Valor máximo de seguridad, momento
rGKraft	REAL	Valor máximo de seguridad, fuerza
bAmpelGruen	BOOL	Enciende la lámpara verde
bAmpelRot	BOOL	Enciende la lámpara roja
uKV	USINT	Cuál sensor de fuerza es utilizado (0: 40kN; 1: 150kN)
bAutoBetrieb	BOOL	Permite iniciar el modo automático
bDatei	BOOL	Indica cuándo se debe guardar los datos
bDirection	USINT	0: Abrir; 1: Cerrar
bEndPosSpeich	BOOL	Graba la posición final si es TRUE
bErrorEnd	BOOL	Muestra cuando hay un error con la posición final
bErrorReset	BOOL	Borra el error del eje
bFileError	BOOL	Indica error en el guardado del archivo de datos
bFileErrorErk	BOOL	Borra el error del guardado de archivo

Tabla 9 – (continuación)

Nombre	Tipo	Descripción
bFileErstellen	BOOL	Indica cuándo se debe crear un nuevo archivo
bHaube1	BOOL	Contacto 1 del interruptor de seguridad
bHaube2	BOOL	Contacto 2 del interruptor de seguridad
bHaubeAuf	BOOL	Indica si la cubierta puede estar abierta
bHome	BOOL	Si es TRUE, se intenta referenciar el eje
bHubSaul	BOOL	Si es TRUE, es posible regular la altura de la mesa
bInduktiv	BOOL	Reconoce las bocas
bKV	BOOL	Indica si la mordaza tiene un amplificador de fuerza
bLeistungEin	BOOL	Activa los relés para tener energía en el sistema
bMove	BOOL	Si es TRUE, se mueve el sistema en el modo manual
bNameEx	BOOL	Indica si el nombre del archivo a crear ya existe
bNeueTest	BOOL	Si es TRUE, se puede iniciar un nuevo test
bNotAus	BOOL	Variable para el controlador de seguridad
bNotAusIn1	BOOL	Contacto 1 del interruptor de emergencia
bNotAusIn2	BOOL	Contacto 2 del interruptor de emergencia
bNotAusOut1	BOOL	Señal para los relés del interruptor de emergencia
bNotAusOut2	BOOL	Señal para los relés del interruptor de emergencia
bParMode	BOOL	Modo de ingreso de parámetros (0: De la lista; 1: Manual)
bPowerOn	BOOL	Si es TRUE, se intenta encender el controlador
bStartPosSpeich	BOOL	Si es TRUE, se guarda la posición inicial
bStepEnable	BOOL	Variable necesaria para permitir la mayoría de procesos

Tabla 9 – (continuación)

Nombre	Tipo	Descripción
bTestAbbrechen	BOOL	Detiene la prueba actual
bVisuPow	BOOL	Indica cuándo el usuario debe ser capaz de apagar el controlador (en la HMI)
bWerteUngueltig	BOOL	Si es TRUE, los parámetros ingresados/calculados no son correctos
cKonfigAnalogIn	BuR_Config_Analog_TYP	Configuración para escalar una entrada analógica
gAxis01	ACP10AXIS_typ	Axis variable
gBasicControlA01	basic_typ	Variable para el control de un eje
iCounter	UDINT	Contador de ciclo
iKraftSensor1	INT	Entrada del sensor de fuerza pequeño
iKraftSensor2	INT	Entrada del sensor de fuerza grande
iMomentSensor	INT	Entrada del sensor de torque
iWinkelKorr	INT	Ángulo de corrección en los modos según momento y fuerza
iWinkelPosReg	INT	Ángulo deseado en el modo de control según posición
rDistance	REAL	Variable para el modo de control según posición
rEndPosition	REAL	Variable para el proceso de cerrar la mordaza en el modo automático
rK2MHolder	REAL	Value Holder
rK2MomentErreicht	REAL	Momento alcanzado en el desacople
rKMHolder	REAL	Value Holder
rKMomentErreicht	REAL	Momento alcanzado en el acople
rKraft40kN	REAL	Kraft Wert (kN)
rKraft150kN	REAL	Valor actual de fuerza (kN)
rKraftErreicht	REAL	Fuerza alcanzada

Tabla 9 – (continuación)

Nombre	Tipo	Descripción
rKraftFiltriert40kN	REAL	Valor filtrado de fuerza (V)
rKraftFiltriert150kN	REAL	Valor filtrado de fuerza (V)
rKraftHolder	REAL	Value holder
rKraftIst40kN	REAL	Valor no filtrado de fuerza
rKraftIst150kN	REAL	Valor no filtrado de fuerza
rMaxKraft	REAL	Valor deseado de fuerza
rMaxMoment	REAL	Valor deseado de momento/torque
rMHolder	REAL	Value holder
rMoment	REAL	Valor actual de momento (Nm)
rMomentErreicht	REAL	Momento alcanzado
rMomentIst	REAL	Valor de momento no filtrado
rOffnungsWeite	REAL	Longitud de apertura
rPosition	REAL	Posición (°)
rPosKup1	REAL	Kupplung Position
rPosKup2	REAL	Posición de desacople
rStartPos	REAL	Variable para la apertura en el modo automático (°)
rStartPosition	REAL	Variable para la apertura en el modo automático (Einheiten–unidades–)
bErrorStateErk	BOOL	Indica si se ha reconocido algún error
bAnschlag	BOOL	Indica si la mordaza tiene (o se ha reconocido que tiene) un tope de torque
sErrorText	STRING[80]	Texto de error
bErrorState	BOOL	Indica si el sistema está en un estado de error

Tabla 9 – (continuación)

Nombre	Tipo	Descripción
rSteigung	REAL	Steigung (mm)
rWHolder	REAL	Value holder
rWinkelErreicht	REAL	Posición alcanzada
sDateiName	STRING[160]	Nombre para el archivo de datos
sHolder	STRING[80]	Value holder
sKonstrukteur	STRING[80]	Nombre del responsable
sVersuch	STRING[80]	Nombre de la prueba
tPZ1	TIME	Delay
tPZ2	TIME	Delay
uHub	UDINT	Número de ciclos
uMod	USINT	Steuerungsart
uOffset	UDINT	Variable para el guardado de datos
uStateMachine	UINT	Estado de la “Máquina de estado”

Fuente: Elaboración propia

Apéndice D
Esquema de conexiones

BESEL & SCHWÄLLER
Schaltanlagenbau GmbH

Besel & Schwäller
Automationstechnik GmbH
Birkenweg 9a
D-87459 Pfronten
Tel.: +49 (0)8363 - 33066 - 75
Fax.: +49 (0)8363 - 33066 - 90
info@besel-schwaeller.de

120

Empresa / Cliente	Allmatik-Jakob Spannsystem GmbH
Descripción	Banco de pruebas
Número	S...
Fabricante Nr.	...

Año de construcción	2017
Voltaje nominal	400 V AC
Corriente nominal	32A
Potencia nominal	11kW
Corriente máxima	32A
	5x4mm²
Funktion	...
Responsable del proyecto	Temiz, Engin
Modificado el	21.04.2017

2

			Datum	03.01.2017	...		Titel- / Deckblatt	Besel & Schwäller Automationstechnik GmbH	=
			Bearb		Teststand Schraubmodul				+
			Gepr						
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch		S...	Blatt 1 / Bl 27

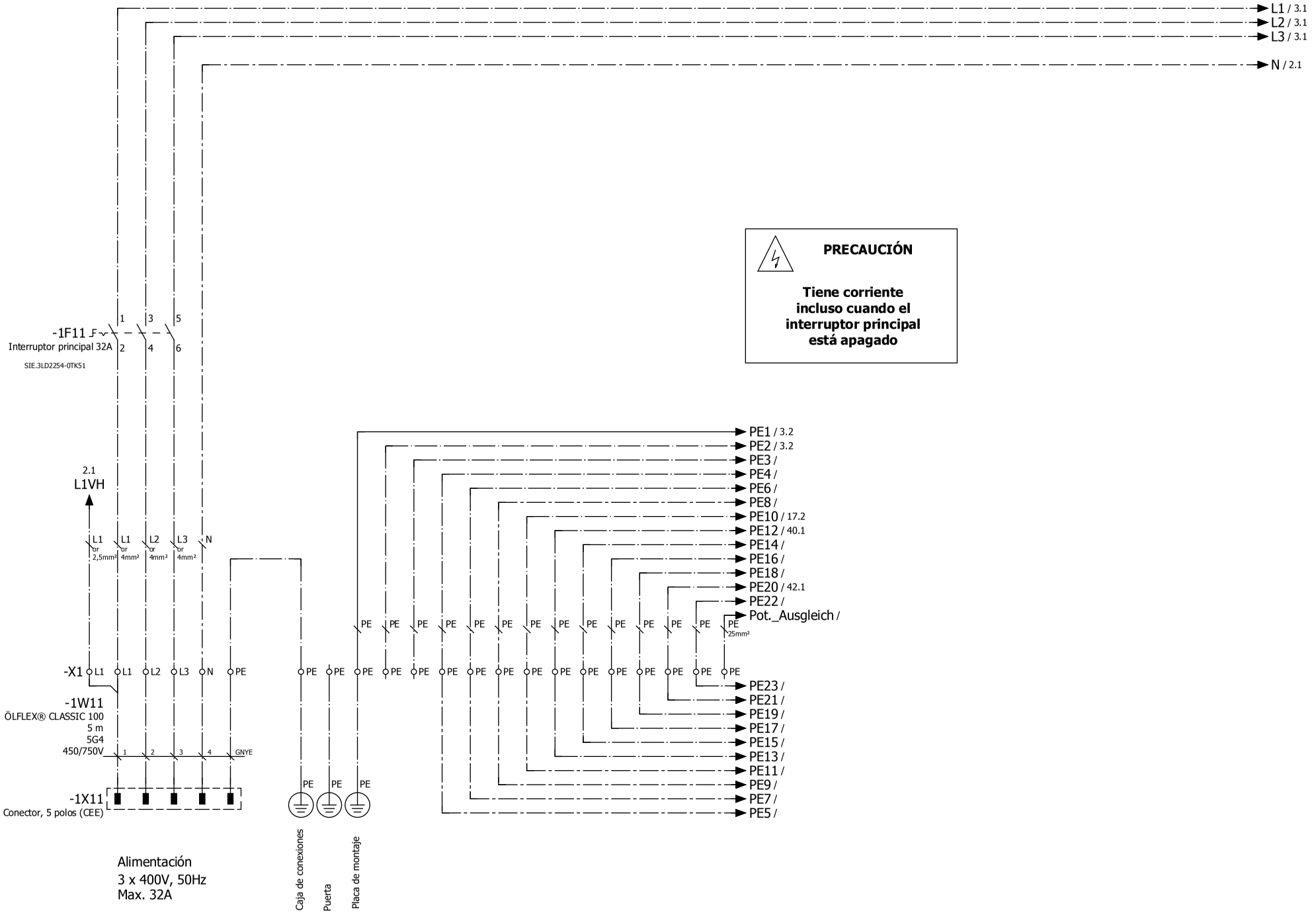
Contenido

F06_001

121

Contenido

F06_001



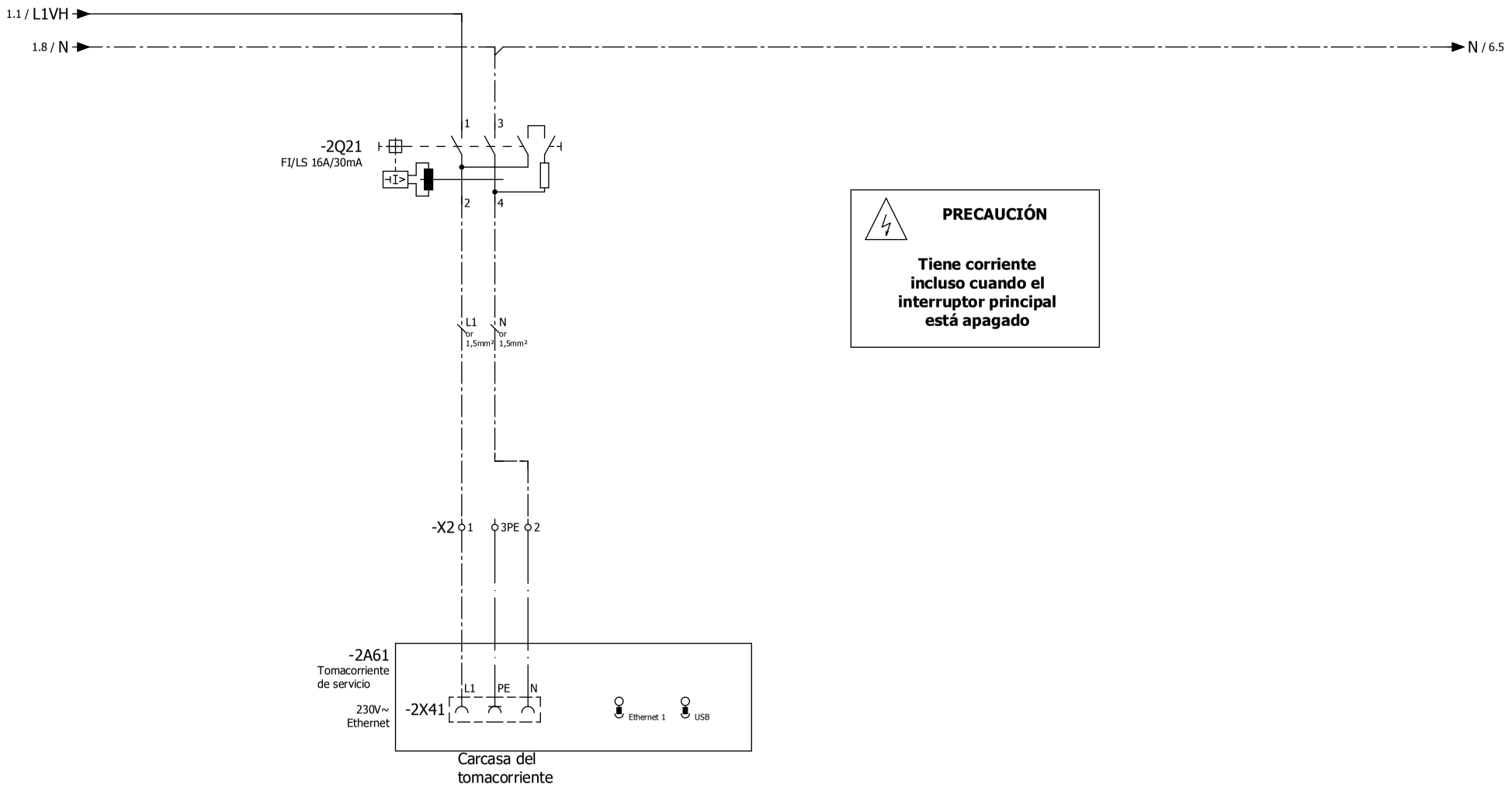
123

=+/3

-X1 = Alimentación 3x400V/N/PE

2

			Datum	03.01.2017	...		Alimentación 400V	Besel & Schwäller Automationstechnik GmbH	= M
			Bearb		Teststand Schraubmodul	BESEL & SCHWÄLLER Schaltanlagenbau GmbH			+ S
			Gepr						
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch		S...	Blatt 1 / Bl 27



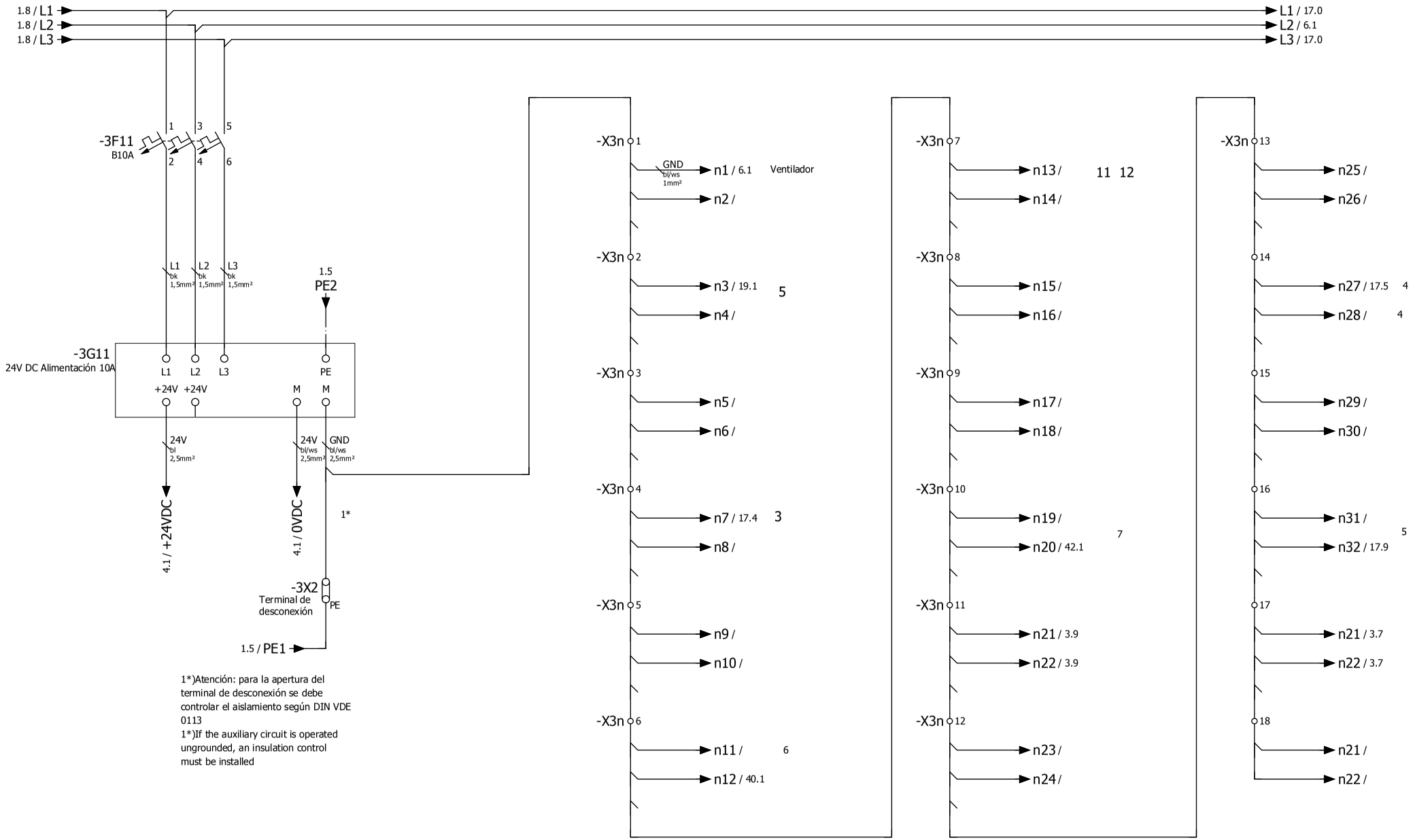
PRECAUCIÓN

Tiene corriente incluso cuando el interruptor principal está apagado

124

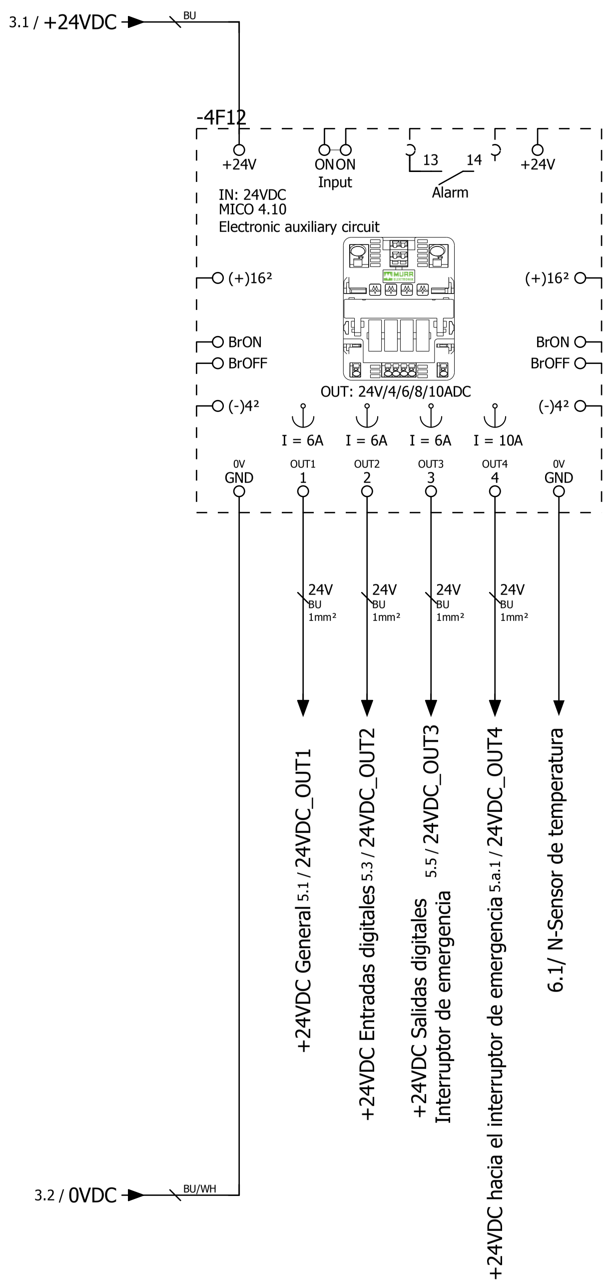
-X2 = Tomacorriente de servicio

			Datum	03.01.2017	...		Tomacorriente de servicio
			Bearb		Teststand Schraubmodul	BESEL & SCHWÄLLER Schaltanlagenbau GmbH	Besel & Schwäller Automationstechnik GmbH = M + S
			Gepr				
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch	S... Blatt 2 Bl 27

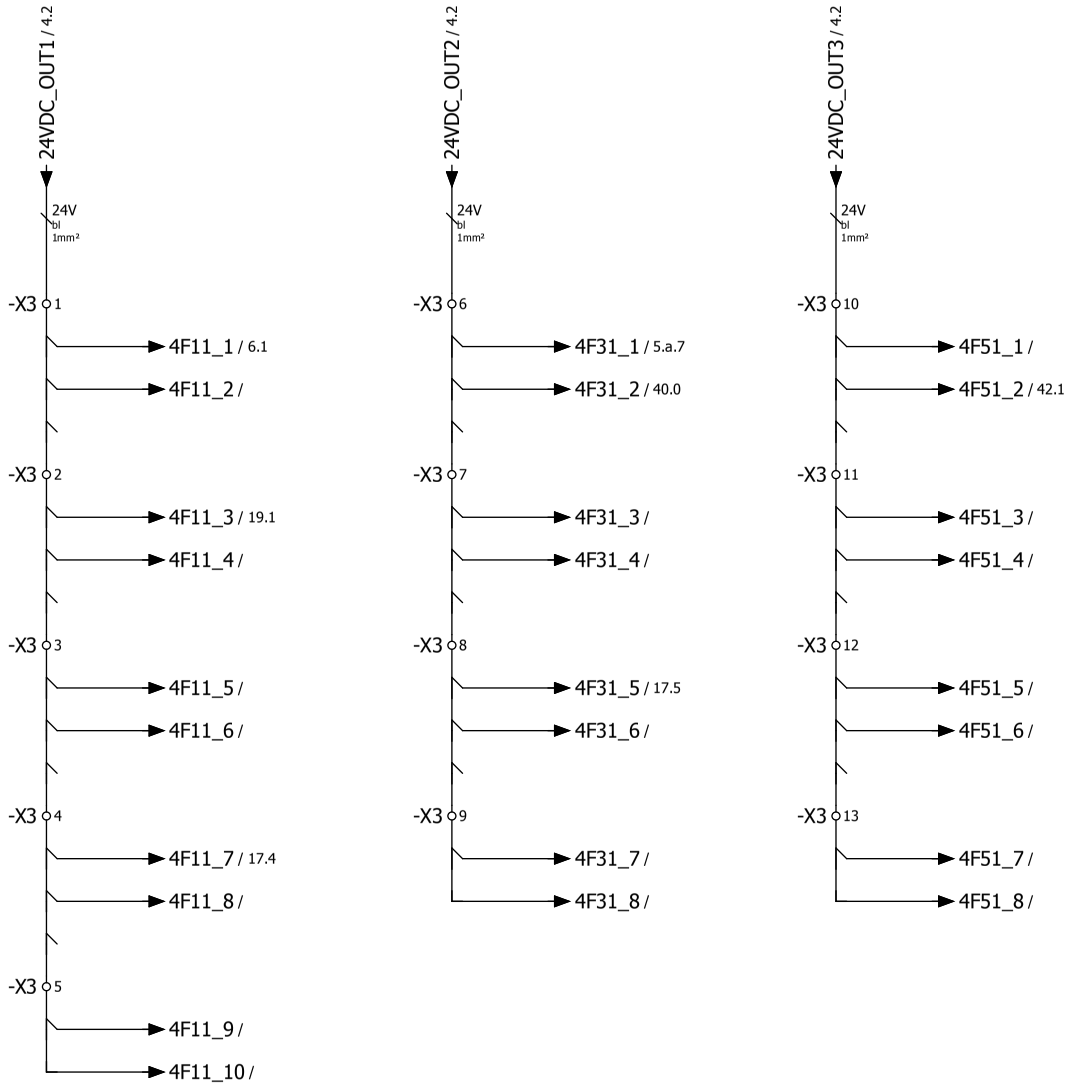


-X3n = Distribución 0VDC

			Datum	03.01.2017	...		Transformador del controlador +24VDC	Besel & Schwäller Automationstechnik GmbH	= M
			Bearb		Teststand Schraubmodul	BESEL & SCHWÄLLER Schaltanlagenbau GmbH			+ S
			Gepr					S...	Blatt 3
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch			Bl 27



			Datum	03.01.2017	...	BESEL & SCHWÄLLER Schaltanlagenbau GmbH	Distribución seguridad +24VDC	Besel & Schwäller Automationstechnik GmbH	= M
			Bearb		Teststand Schraubmodul				+ S
			Gepr					S...	Blatt 4
Änderung	Datum	Name	Urspr		Ersatz von / Ersetzt durch				Bl 27



+24VDC General

+24VDC Entradas digitales

+24VDC Salidas digitales del interruptor de emergencia

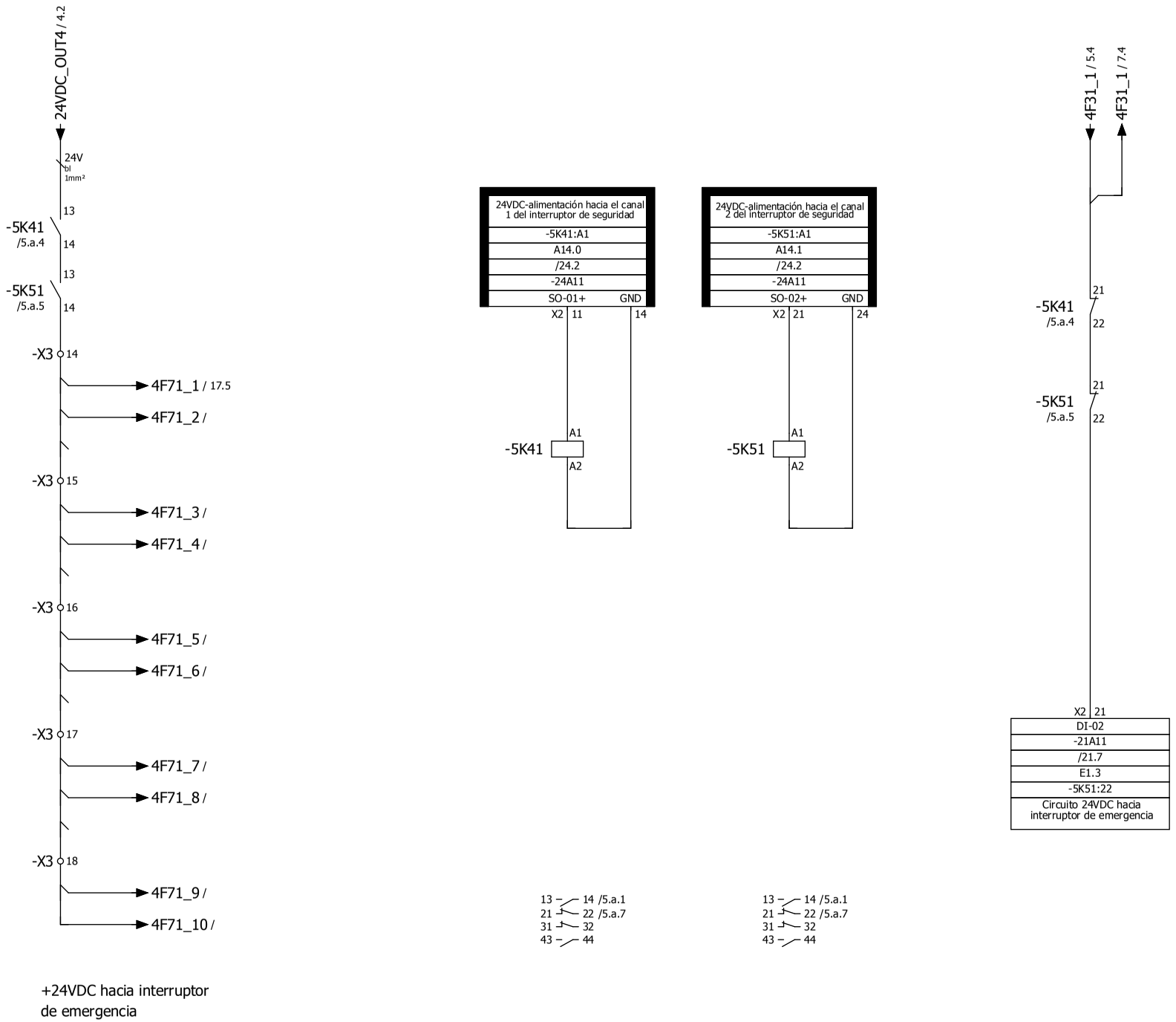
127

-X3 = Distribución +24VDC

5.a

4

			Datum	03.01.2017	...		Distribución +24VDC	Besel & Schwäller Automationstechnik GmbH	= M
			Bearb		Teststand Schraubmodul	BESEL & SCHWÄLLER Schaltanlagenbau GmbH			+ S
			Gepr						
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch		S...	Blatt 5 / Bl 27



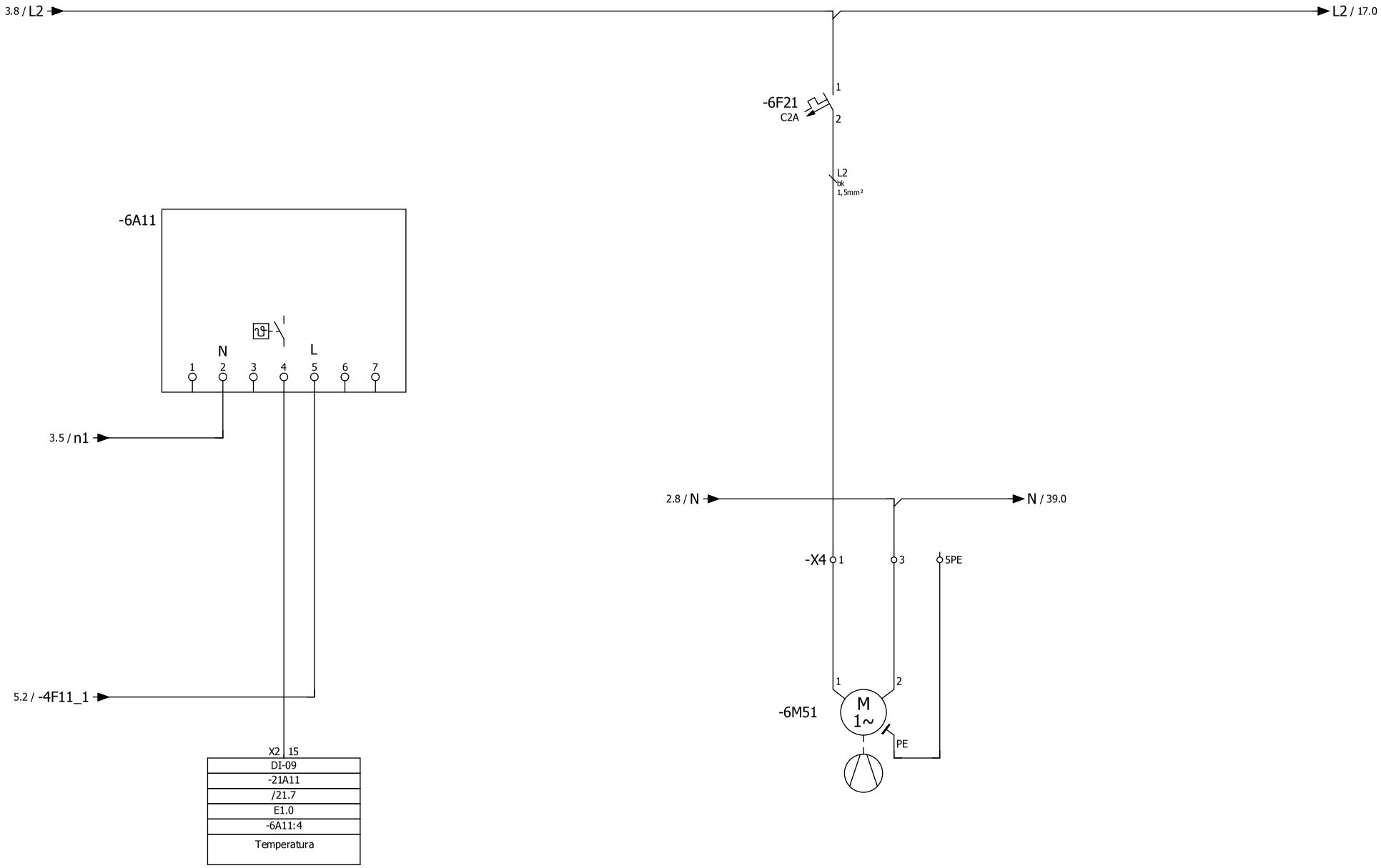
128

5

-X3 = Distribución +24VDC

6

			Datum	03.01.2017	...		Distribución +24VDC	Besel & Schwäller Automationstechnik GmbH	= M
			Bearb		Teststand Schraubmodul	BESEL & SCHWÄLLER Schaltanlagenbau GmbH			+ S
			Gepr						
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch		S...	Blatt 5.a
									Bl 27



Ventilador izquierda

129

5.a

-X4 = Ventilador 7

			Datum	03.01.2017	...		Ventilador	Besel & Schwäller Automationstechnik GmbH	= M
			Bearb		Teststand Schraubmodul	BESEL & SCHWÄLLER Schaltanlagenbau GmbH			+ S
			Gepr						
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch		S...	Blatt 6 / Bl 27

4F31_1 / 5.a.7
4F31_1 / 39.8

X1	14
DI-01	
-21A11	
/21.7	
E0.0	
+M-7S11:34	
Contacto 1 interruptor de emergencia	

X1	15
DI-03	
/21.7	
E0.1	
-X7:13	
Contacto 2 interruptor de emergencia	

BU

-24A11	Pulse-01SI-01	X1 12
/24.2		X1 11
A10.0		
+M-7S11:11		
Not-Aus Kreis Kanal 1		

-24A11	Pulse-03SI-03	X1 14
/24.2		X1 13
A10.1		
-24A11:X1:14		
Not-Aus Kreis Kanal 2		

-X7 1 2 3 4 5 6 7PE 8 9 10 11 12 13 14PE

-7W21
ÖLFLEX CLASSIC 110
7G0,75
1 2 3 4 5 6 GNYE

+M

-7S11

-PE

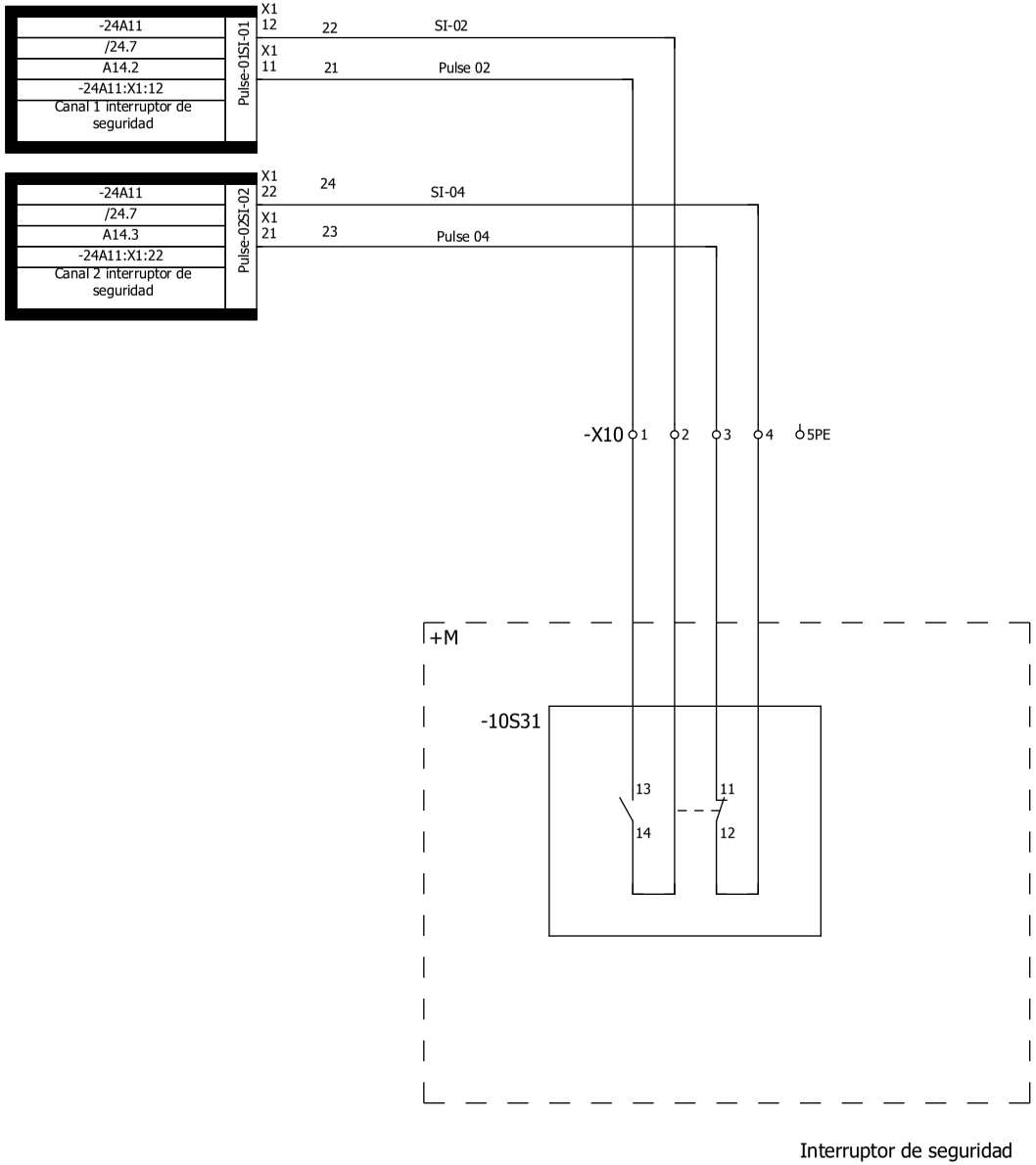
11 21 33
12 /7.3 22 /7.3 34

Interruptor de emergencia
Automation

Interruptor de emergencia
Reserve

-X7 = circuito interno del interruptor de emergencia

			Datum	03.01.2017	...	Circuito interno del interruptor de emergencia	Besel & Schwäller Automationstechnik GmbH
			Bearb		Teststand Schraubmodul		= M + S
			Gepr				
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch	S... Blatt 7 Bl 27

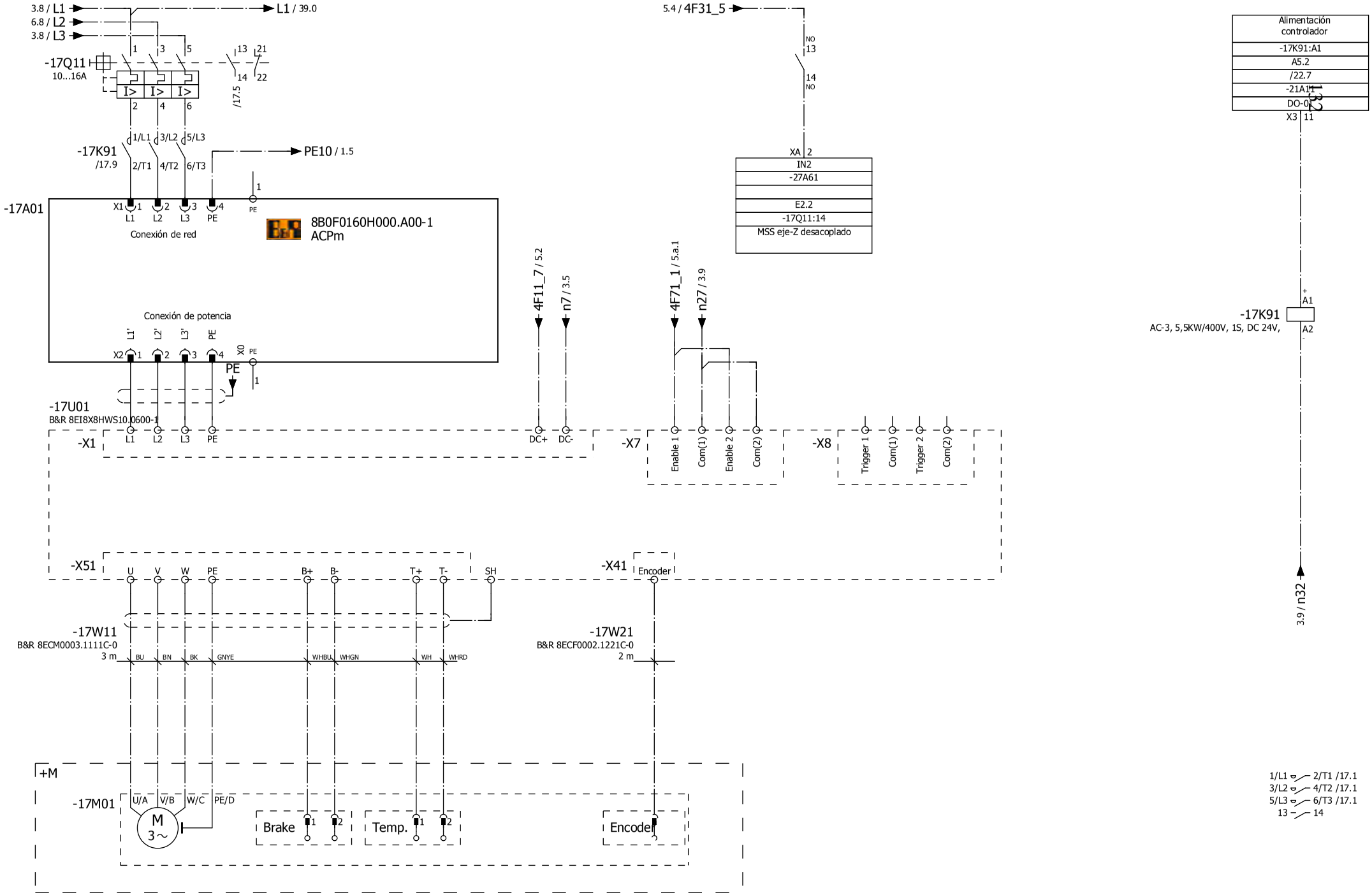


Interruptor de seguridad

131

-X10 = interruptor de seguridad de lla cubierta

			Datum	03.01.2017	...		Interruptor de seguridad de la cubierta	Besel & Schwäller Automationstechnik GmbH	= M
			Bearb		Teststand Schraubmodul	BESEL & SCHWÄLLER Schaltanlagenbau GmbH			+ S
			Gepr						
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch		S...	Blatt 10 / Bl 27



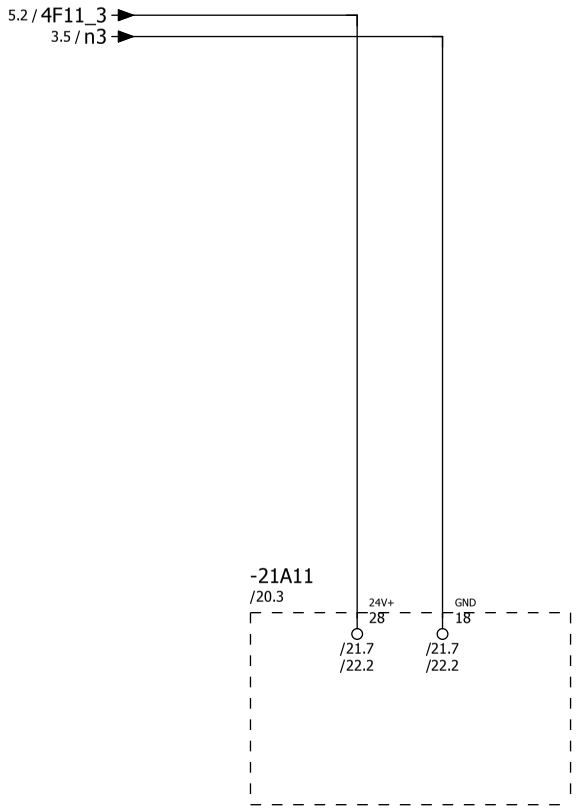
10

-17X3 = freno eje-Z

19

			Datum	03.01.2017	...		Servocontrolador eje-Z	Besel & Schwäller Automationstechnik GmbH
			Bearb		Teststand Schraubmodul			= M
			Gepr					+ S
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch		S...

Blatt 17
Bl 27



133

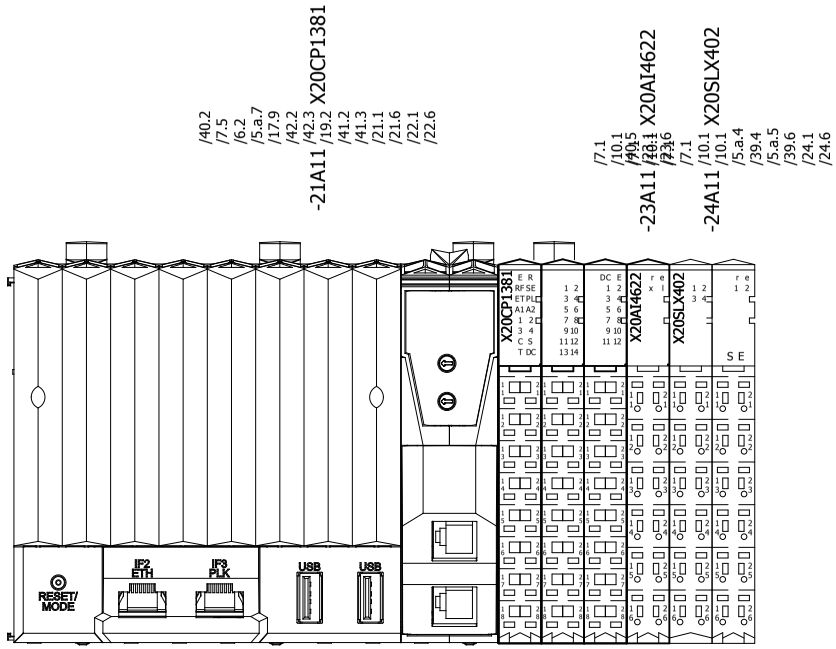
17

20

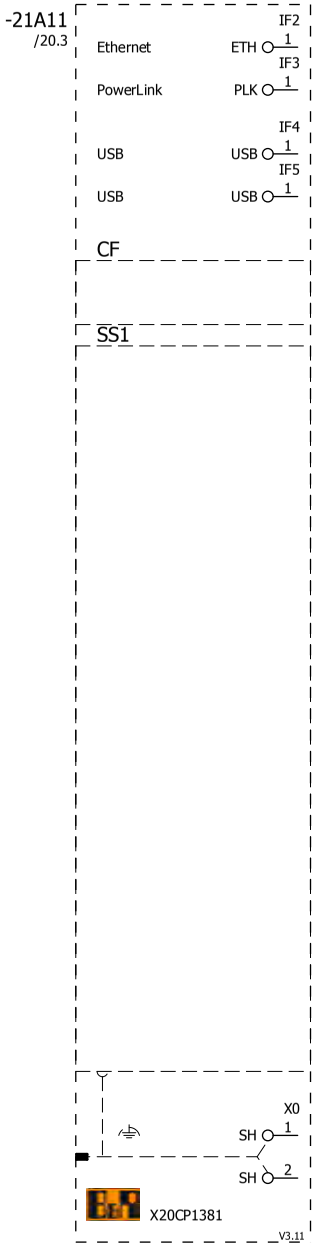
			Datum	03.01.2017	...		Alimentación CPU	Besel & Schwäller Automationstechnik GmbH	= M
			Bearb		Teststand Schraubmodul	BESEL & SCHWÄLLER Schaltanlagenbau GmbH			+ S
			Gepr						
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch		S...	Blatt 19 / Bl 27

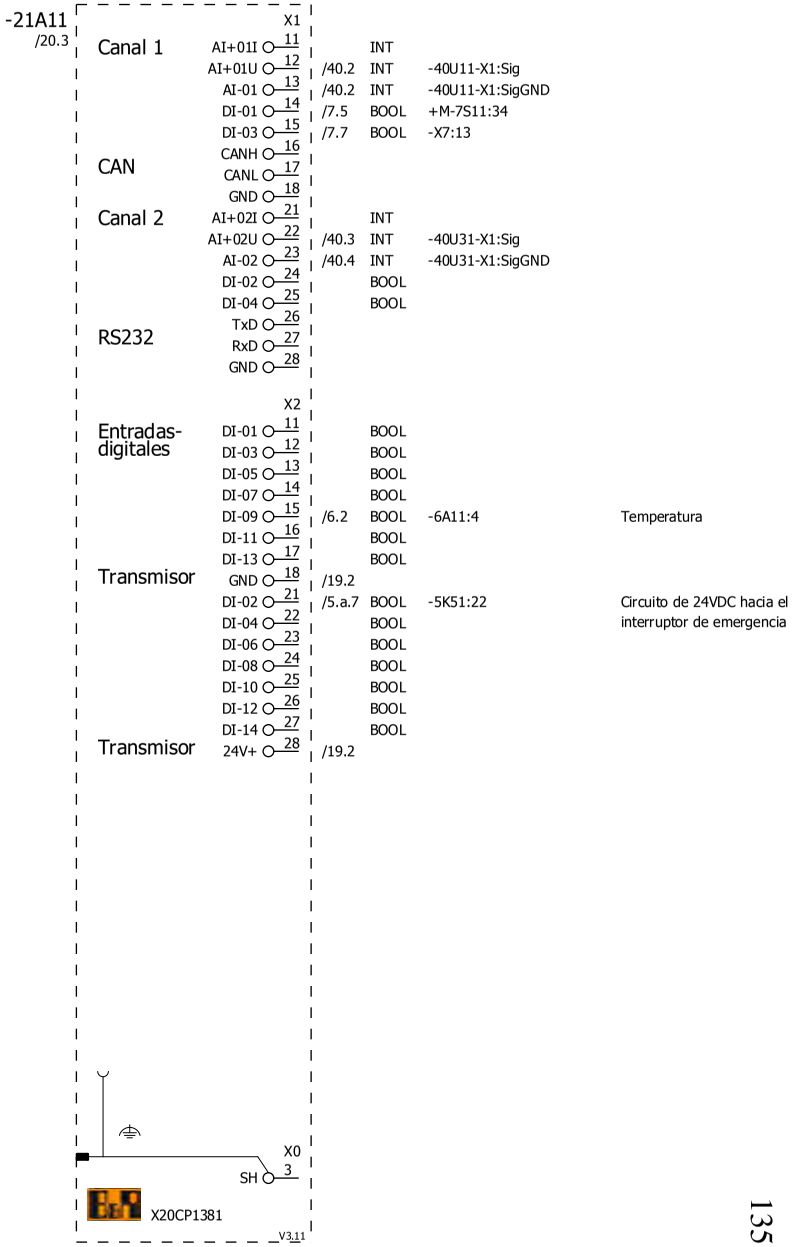
B&R Controller X20 Compact CPU

X20CP1381



			Datum	03.01.2017	...		Controlador del PLC	Besel & Schwäller Automationstechnik GmbH	= M
			Bearb		Teststand Schraubmodul	BESEL & SCHWÄLLER Schaltanlagenbau GmbH			+ S
			Gepr						
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch		S...	Blatt 20 Bl 27





135

20

22

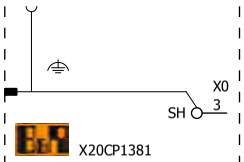
			Datum	03.01.2017	...	Unidad de CPU, PC, PLC	Besel & Schwäller Automationstechnik GmbH
			Bearb		Teststand Schraubmodul		= M
			Gepr				+ S
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch	S...

BESEL & SCHWÄLLER Schaltanlagenbau GmbH

Blatt 21
Bl 27

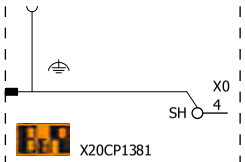
-21A11
/20.3

	X1					
Canal 1	n.c.	11				
	Sensor+01	12		INT		
	Sensor-01	13		INT		
	DI-01	14	/41.2	BOOL	-X41:11S	Digital Input 3.0
	DI-03	15	/41.3	BOOL	-X41:12S	Digital Input 3.0
CAN	CANH	16				
	CANL	17				
	GND	18				
Canal 2	AI+02I	21		INT		
	AI+02U	22		INT		
	AI-02	23		INT		
	DI-02	24	/41.4	BOOL	-X41:13S	Digital Input 3.2
	DI-04	25	/41.5	BOOL	-X41:14S	Digital Input 3.3
RS232	TxD	26				
	RxD	27				
	GND	28				
	X2					
Entradas-digitales	DI-01	11	/41.6	BOOL	-X41:15S	Digital Input 3.4
	DI-03	12		BOOL		
	DI-05	13	/41.8	BOOL	-X41:17S	Digital Input 3.6
	DI-07	14		BOOL		
	DI-09	15		BOOL		
	CNT-01	16		DINT		
	DI-13	17		BOOL		
Transmisor	GND	18	/19.2			
	DI-02	21		BOOL		
	DI-04	22		BOOL		
	DI-06	23		BOOL		
	DI-08	24		BOOL		
	DI-10	25		BOOL		
	DI-12	26		BOOL		
	DI-14	27		BOOL		
Transmisor	24V+	28	/19.2			



-21A11
/20.3

	X3					
Salidas-digitales	DO-01	11	/17.9	BOOL	-17K91:A1	Alimentación controlador
	DO-03	12		BOOL		
	DO-05	13		BOOL		
	DO-07	14		BOOL		
	DO-09	15	/42.2	BOOL	-X42:2S	Digital Output 3.0
	DO-11	16	/42.3	BOOL	-X42:3S	Digital Output 3.0
	VDC	17				
	GND	18				
	DO-02	21	/42.4	BOOL	-X42:4S	Digital Output 3.2
	DO-04	22	/42.5	BOOL	-X42:5S	Digital Output 3.3
	DO-06	23	/42.6	BOOL	-X42:6S	Digital Output 3.4
	DO-08	24	/42.7	BOOL	-X42:7S	Digital Output 3.5
	DO-10	25	/42.8	BOOL	-X42:8S	Digital Output 3.6
	DO-12	26	/42.8	BOOL	-X42:9S	Digital Output 3.5
	VDC	27				
	GND	28				



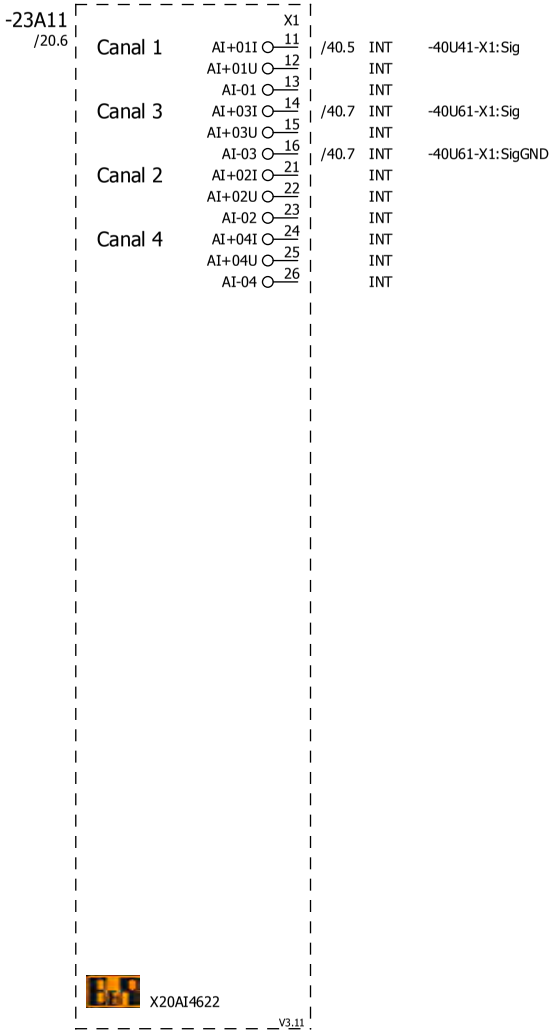
21

23

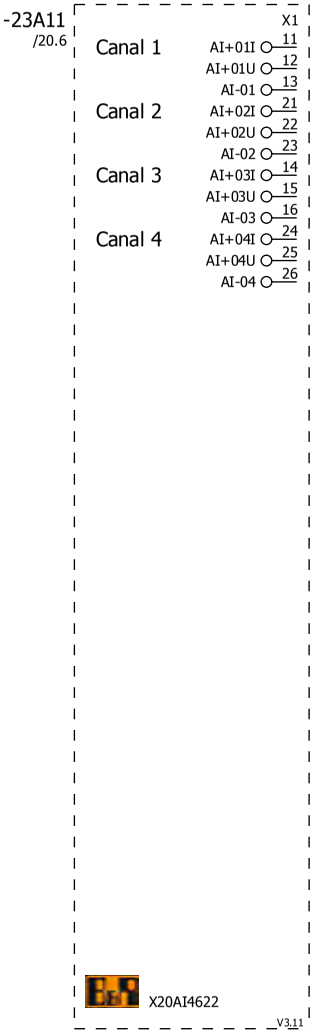
			Datum	03.01.2017	...	Unidad de CPU, PC, PLC	Besel & Schwäller Automationstechnik GmbH
			Bearb		Teststand Schraubmodul		= M
			Gepr				+ S
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch	S...

BESEL & SCHWÄLLER Schaltanlagenbau GmbH

Blatt 22
Bl 27



/40.5	INT	-40U41-X1:Sig
	INT	
	INT	
/40.7	INT	-40U61-X1:Sig
	INT	
/40.7	INT	-40U61-X1:SigGND
	INT	
	INT	
	INT	
	INT	
	INT	
	INT	



INT
INT
INT
INT
INT
INT
INT
INT
INT
INT
INT
INT

137

			Datum	03.01.2017	...	PLC-Analogicos I	Besel & Schwäller Automationstechnik GmbH
			Bearb		Teststand Schraubmodul		= M + S
			Gepr				
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch	S... Blatt 23 Bl 27

-24A11 /20.6

Conector iquierda — X1

Pulse-01	11	/7.1	BOOL	+M-7S11:11	Circuito de interruptor de emergencia canal 1
SI-01	12	/7.1	BOOL	-24A11:X1:11	
Pulse-03	13	/7.1	BOOL	-24A11:X1:14	Circuito de interruptor de emergencia canal 2
SI-03	14	/7.1	BOOL	+M-7S11:22	
n.c.	15		BOOL		
n.c.	16		BOOL		
Pulse-02	21		BOOL		
SI-02	22		BOOL		
Pulse-04	23		BOOL		
SI-04	24		BOOL		
n.c.	25		BOOL		
n.c.	26		BOOL		

Conector derecha — X2

SO-01+	11	/5.a.4	BOOL	-5K41:A1	24VDC-alimentación hacia canal 1 interruptor de emergencia
n.c.	12		BOOL		
n.c.	13		BOOL		
GND	14	/39.5	BOOL		
GND	15	/5.a.4	BOOL		
GND	16		BOOL		
SO-02+	21	/5.a.5	BOOL	-5K51:A1	24VDC-alimentación hacia cancal 2 interruptor de emergencia
n.c.	22		BOOL		
n.c.	23		BOOL		
GND	24	/5.a.6	BOOL		
GND	25	/39.6	BOOL		
GND	26		BOOL		

X20
X20SLX402
V3.11

X20 safe digital mixed module, safety controller, openSAFETY, 11 openSAFETY nodes, 4 SafeMOTION axes, 4 safe digital inputs, configurable input filter, 4 pulse outputs, 24 VDC, 2 safe digital type B outputs, 24 VDC, 0.2 A, OSSD <10 µs

-24A11 /20.6

Conector izquierda — X1

Pulse-01	11	/10.1	BOOL	-24A11:X1:12	Canal 1 interruptor de seguridad
Pulse-02	21	/10.1	BOOL	-24A11:X1:22	Canal 2 interruptor de seguridad
SI-01	12	/10.1	BOOL	-X10:2	Canal 1 interruptor de emergencia
SI-02	22	/10.1	BOOL	-X10:4	Canal 2 interruptor de emergencia
Pulse-03	13		BOOL		
Pulse-04	23		BOOL		
SI-03	14		BOOL		
SI-04	24		BOOL		
n.c.	15		BOOL		
n.c.	25		BOOL		
n.c.	16		BOOL		
n.c.	26		BOOL		

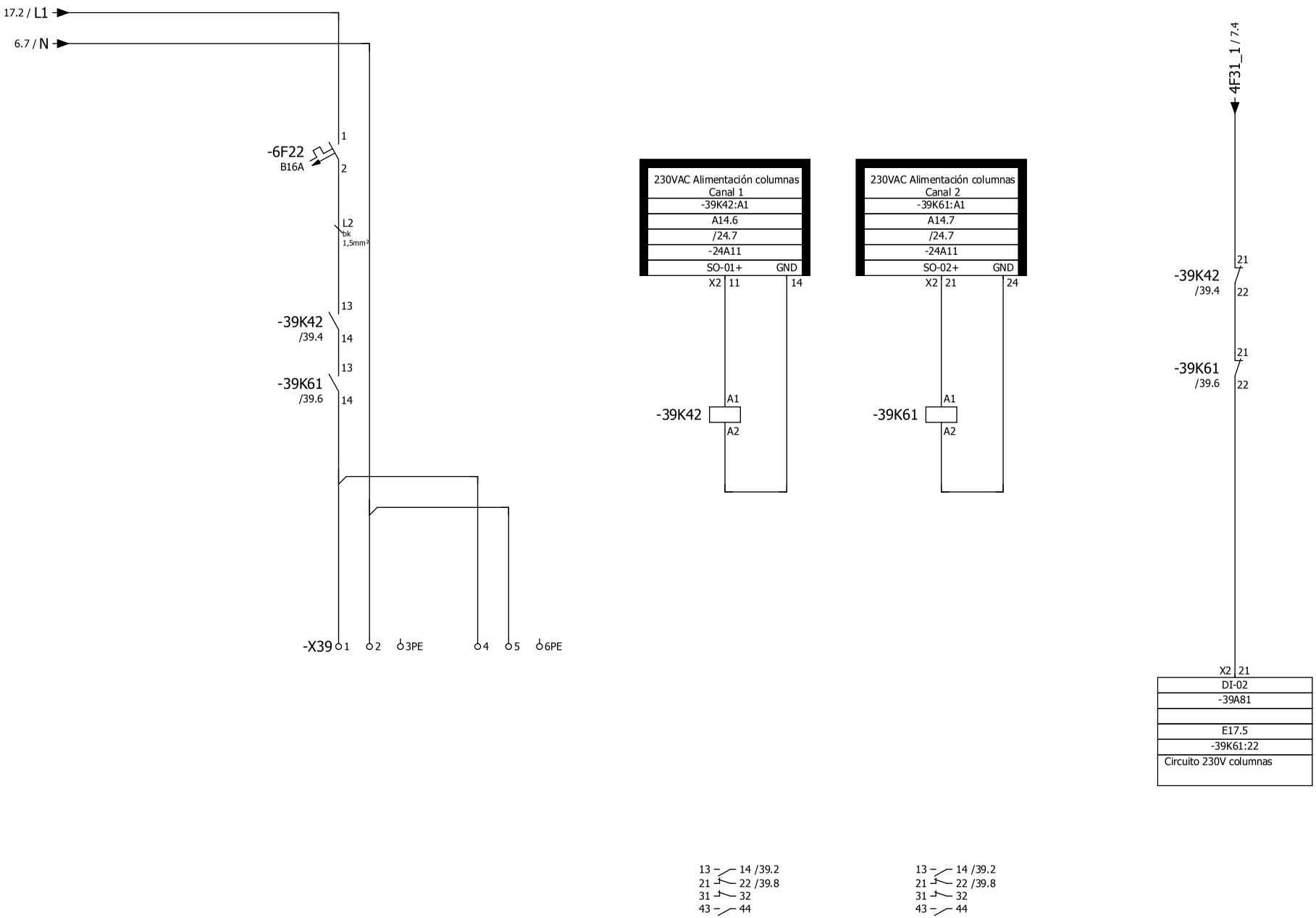
Conector derecha — X2

SO-01+	11	/39.4	BOOL	-39K42:A1	230VAC Alimentación columnas canal 1
SO-02+	21	/39.6	BOOL	-39K61:A1	230VAC Alimentación columnas canal 2
n.c.	12		BOOL		
n.c.	22		BOOL		
n.c.	13		BOOL		
n.c.	23		BOOL		
GND	14	/39.5	BOOL		
GND	24	/5.a.6	BOOL		
GND	15	/39.6	BOOL		
GND	25		BOOL		
GND	16		BOOL		
GND	26		BOOL		

X20
X20SLX402
V3.11

X20 safe digital mixed module, safety controller, openSAFETY, 11 openSAFETY nodes, 4 SafeMOTION axes, 4 safe digital inputs, configurable input filter, 4 pulse outputs, 24 VDC, 2 safe digital type B outputs, 24 VDC, 0.2 A, OSSD <10 µs

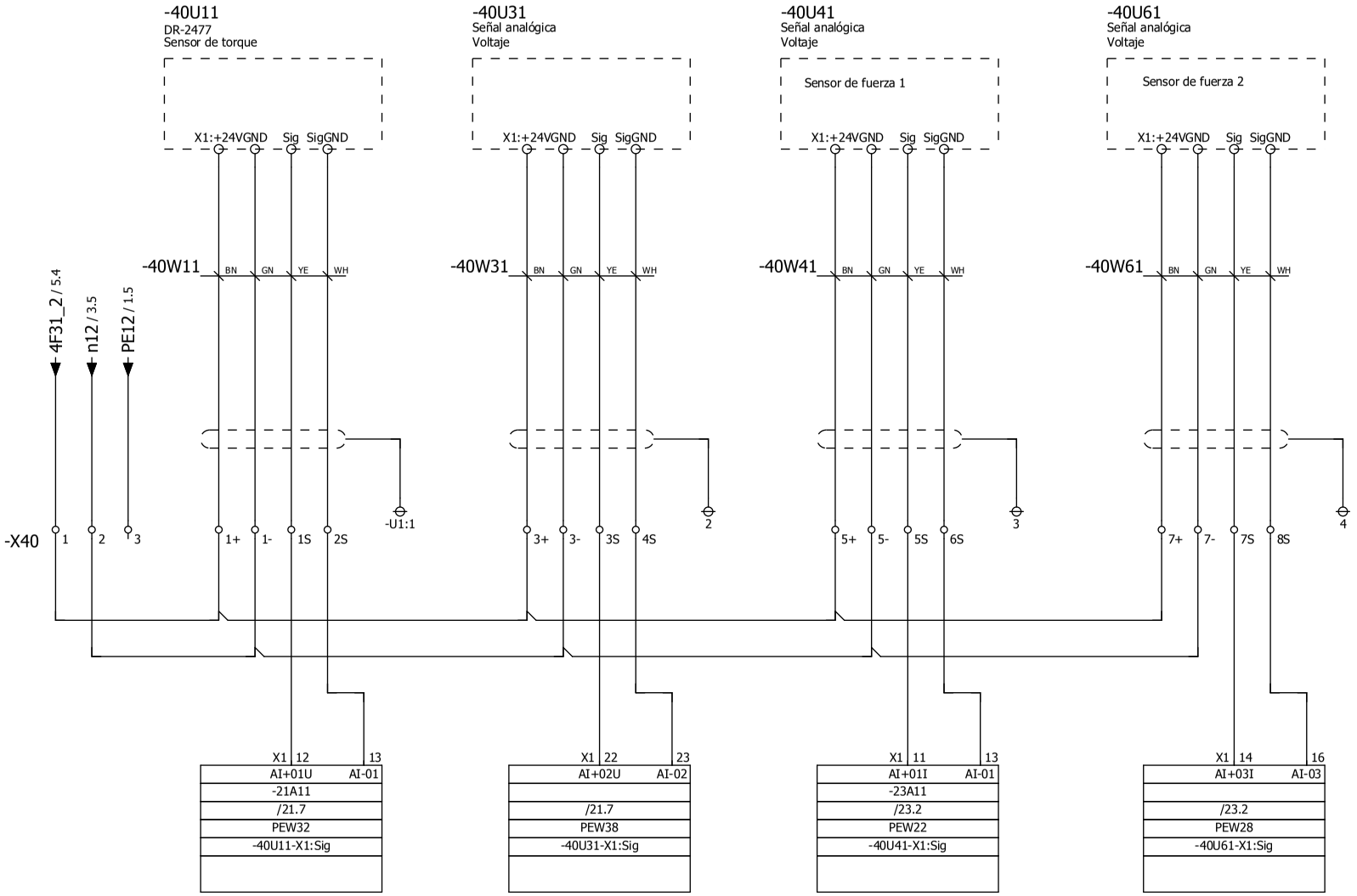
			Datum	03.01.2017	...		PLC-Safety I/O	Besel & Schwäller Automationstechnik GmbH	= M
			Bearb		Teststand Schraubmodul	BESEL & SCHWÄLLER Schaltanlagenbau GmbH			+ S
			Gepr						
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch		S...	Blatt 24 / Bl 27



139

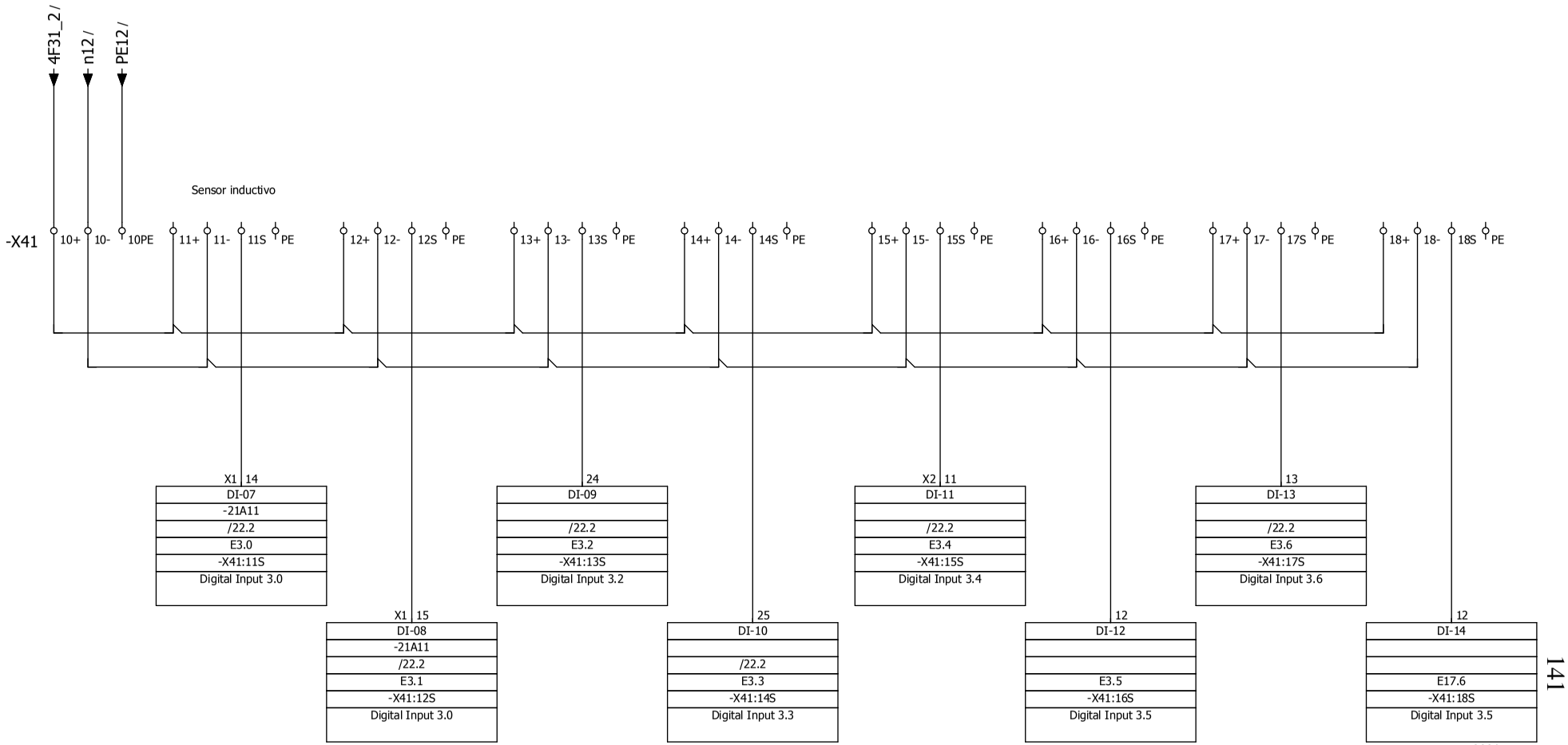
-X39 = tomacorrientes de servicio

			Datum	03.01.2017	...	Alimentación columnas	Besel & Schwäller Automationstechnik GmbH	= M
			Bearb		Teststand Schraubmodul			+ S
			Gepr					
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch	S...	Blatt 39
								Bl 27



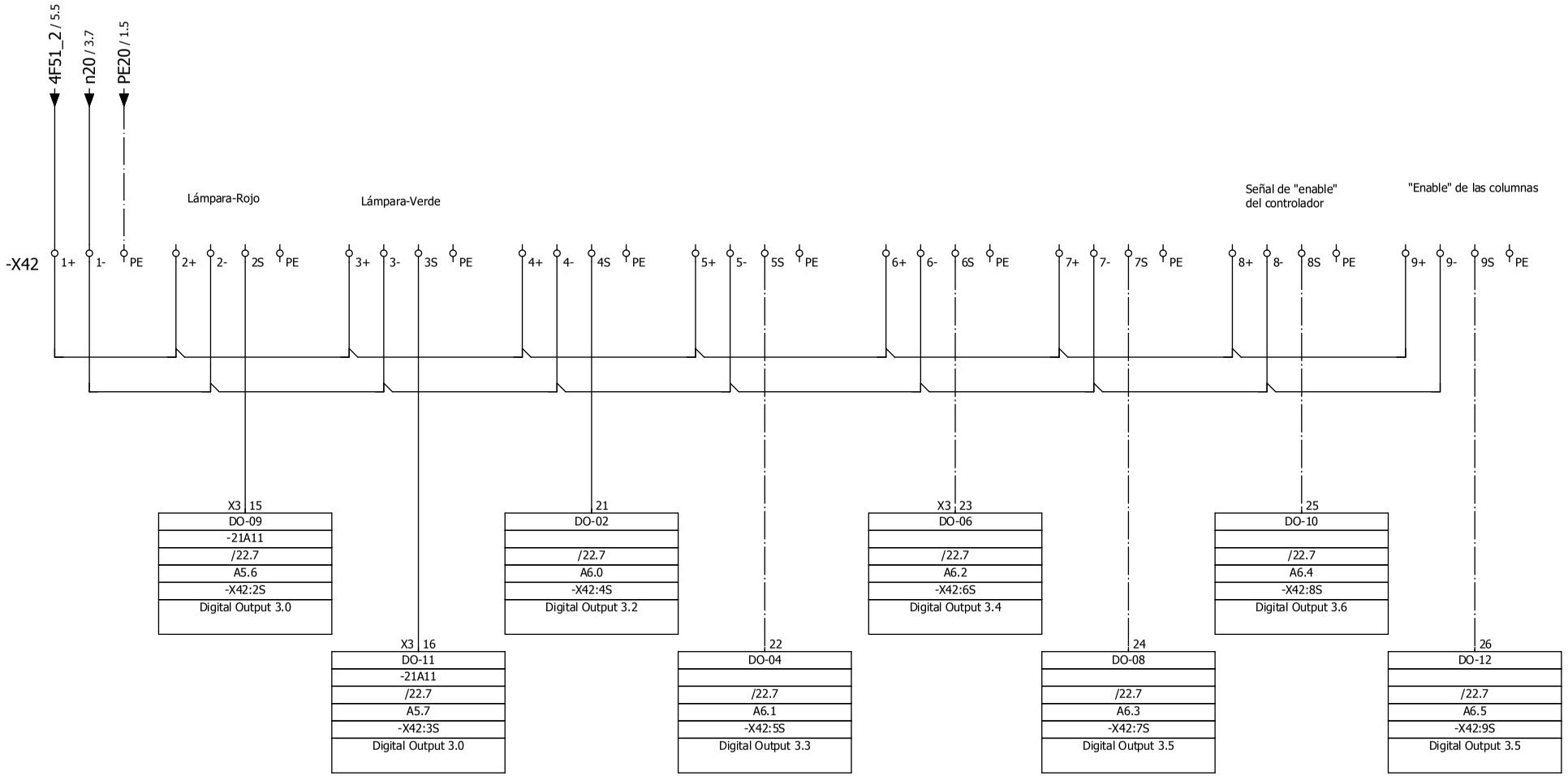
-X40 = Tomacorriente de servicio

			Datum	03.01.2017	...		Interfaz entradas analógicas	Besel & Schwäller Automationstechnik GmbH	= M
			Bearb		Teststand Schraubmodul				+ S
			Gepr						
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch		S...	Blatt 40
									Bl 27



			Datum	03.01.2017	...		Interfaz entradas digitales	Besel & Schwäller Automationstechnik GmbH	= M
			Bearb		Teststand Schraubmodul	BESEL & SCHWÄLLER Schaltanlagenbau GmbH			+ S
			Gepr						
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch		S...	Blatt 41 / Bl 27

142



41

KL+/1

			Datum	03.01.2017	...	Interfaz salidas digitales	Besel & Schwäller Automationstechnik GmbH
			Bearb		Teststand Schraubmodul		= M
			Gepr				+ S
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch	S...

BESEL & SCHWÄLLER Schaltanlagenbau GmbH

Blatt 42
Bl 27

Resumen del terminal de conexiones

F14_001_BS

Conexión	Definición	Terminal					Página de gráfica
		primero	último	Suma PE	Suma N	Total	
				0	0	5	
=M+S-X1	Alimentación 3x400V/N/PE	PE	PE	28	6	38	
=M+S-X2	Tomacorriente de servicio	1	3PE	2	1	6	
=M+S-X3	Distribución +24VDC	1	18	0	0	54	
=M+S-X3n	Distribución 0VDC	1	18	0	0	18	
=M+S-X4	Ventilador	1	5PE	3	2	9	
=M+S-X7	Terminales internos del interruptor de emergencia	1	14PE	4	0	28	
=M+S-X10	Terminales externos del interruptor de emergencia	1	5PE	2	0	10	
=M+S-X39	Tomacorriente de servicio	1	6PE	4	2	12	
=M+S-X40	=	1	8S	9	0	54	
=M+S-X41	=	10+	PE	9	0	70	
=M+S-X42	=	1+	PE	9	0	70	
=M+S-17X3	Terminal del freno del eje-Z			0	0	2	

=M+S/42

=ST/1

			Datum		...	BESEL & SCHWÄLLER Schaltanlagenbau GmbH	Resumen del terminal de conexiones
			Bearb		Teststand Schraubmodul		Besel & Schwäller Automationstechnik GmbH
			Gepr				= KL
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch	+
							S...
							Blatt 1
							Bl 27

Lista de componentes

F01_003BS

Número de artículo	Cantidad	Descripción	Código	Fabricante	BMK	Posición
SIE.3SB3801-0EF3	1	Carcasa para interruptor de emergencia-amarillo	3SB3801-0EF3	SIEMENS	-7S11	=M+S/7.3
SIE.3SB3400-0B	1	Switch,1S,conexión con perno	3SB3400-0B	SIEMENS	-7S11	=M+S/7.3
MUR.4000-68000-1300000	1	Modlink MSDD Conector	4000-68000-1300000	MURR	-2A61	=M+S/2.2
MUR.4000-68522-0000001	1	Placa frontal de carcasa-transparente	4000-68522-0000001	MURR	-2A61	=M+S/2.2
MUR.4000-68000-0020000	1	Modlink MSDD amarillo	4000-68000-0020000	MURR	-2A61	=M+S/2.2
MUR.4000-68000-0020000	1	Modlink MSDD conector amarillo	4000-68000-0020000	MURR	-2X41	=M+S/2.3
MUR.4000-68000-1300000	1	Modlink MSDD conector	4000-68000-1300000	MURR	-2X41	=M+S/2.3
MUR.4000-68522-0000001	1	Placa frontal de carcasa-transparente	4000-68522-0000001	MURR	-2X41	=M+S/2.3
RIT.SK3110.000	1	Termostato	SK 3110.000	RITTAL	-6A11	=M+S/6.1
BuR.8B0F0160H000.A00-1	1	ACPmulti NF 16A 480V	8B0F0160H000.A00-1	BuR	-17A01	=M+S/17.0
BuR.X20CP1381	1	X20 CPU 200, INT.EA ETH PLK CAN	X20CP1381	BuR	-21A11	=M+S/20.3
BuR.X20AI4622	0	X20 Analog 4xE, +/-10V/0..20 mA, 12 Bit	X20AI4622	BuR	-23A11	=M+S/20.6
BuR.X20SLX402	0	X20 SafeLOGIC, 4xE, 24V, 2xA, 200mA, 24V	X20SLX402	BuR	-24A11	=M+S/20.6
SIE.3LD2254-0TK51	1	HPTSCH 3-POLIG 32A/690V 400V/11,5KW	3LD2254-0TK51	SIEMENS	-1F11	=M+S/1.1
SIE.5SY4310-6	1	Relé de seguridad 400V 10KA, 3POLIG, B, 10A, T=70MM	5SY4310-6	SIE	-3F11	=M+S/3.1
MURR.9000-41034-0401000	1	MICO Regulador de corriente, 4-canales	MICO 4.10	MURR	-4F12	=M+S/4.1
SIE.5SJ4102-7HG40	1	Relé de seguridad 2A C	5SJ4102-7HG40	SIEMENS	-6F21	=M+S/6.5
SIE.5SJ4 116-6HG40	1	Relé de seguridad 16A B	5SJ4 116-6HG40	SIEMENS	-6F22	=M+S/39.2
BLO.B0710035	1	Dispositivo de red DMG 400V/24V, 10A	B0710035	BLOCK	-3G11	=M+S/3.1
SIE.3RH2122-1BB40	1	Relé auxiliar, 2S+2OE DC 24V	3RH2122-1BB40	SIE	-5K41	=M+S/5.a.4
SIE.3RH2122-1BB40	1	Relé auxiliar, 2S+2OE DC 24V	3RH2122-1BB40	SIE	-5K51	=M+S/5.a.5
SIE.3RT2017-2BB41	1	Protección, AC-3, 5,5KW/400V, 1S, DC 24V,	3RT2017-2BB41	SIE	-17K91	=M+S/17.9
SIE.3RT1026-1BB40	1	Protección AC3:11KW/400V DC24V	3RT1026-1BB40	SIEMENS	-17K91	=M+S/17.9
SIE.3RH2122-1BB40	1	Relé auxiliar, 2S+2OE DC 24V	3RH2122-1BB40	SIE	-39K42	=M+S/39.4
SIE.3RH2122-1BB40	1	Relé auxiliar, 2S+2OE DC 24V	3RH2122-1BB40	SIE	-39K61	=M+S/39.6
RIT.SK3238.100	1	Filtro del ventilador	SK3238.100	RITTAL	-6M51	=M+S/6.5
RIT.SK3238.200	1	Ventilador	SK3238.200	RITTAL	-6M51	=M+S/6.5
SIE.FI/LS-Schalter 30mA 2polig 16A	1	Protección de exceso de corriente	5SU1 354-6KK16	SIEMENS	-2Q21	=M+S/2.3
SIE.3RV1021-4BA15	1	Relé de seguridad 7,5kW, 14..20A, BGR. S0	3RV1021-4BA15	SIEMENS	-17Q11	=M+S/17.1
	0				-1W11	=M+S/1.1
LAPP.1119107	1	ÖLFLEX CLASSIC 110 7G0,75	ÖLFLEX CLASSIC 110	LAPP	-7W21	=M+S/7.3
	0				-40W11	=M+S/40.1
	0				-40W31	=M+S/40.3
	0				-40W41	=M+S/40.4
	0				-40W61	=M+S/40.6
PXC.3211757	4	Terminal	PT 4	PXC	-X1	=M+S/1.8
PXC.3211760	1	Terminal	PT 4 BU	PXC	-X1	=M+S/1.8
PXC.3030420	1	Tapa cobertora	D-ST 4	PXC	-X1	=M+S/1.8

144

=KL/1

1.a

			Datum		...		
			Bearb		Teststand Schraubmodul		
			Gepr				
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch	

BESEL & SCHWÄLLER Schaltanlagenbau GmbH

Lista de componentes : SIE.3SB3801-0EF3 - PXC.3030420

Besel & Schwäller Automationstechnik GmbH

= ST

+

S...

Blatt 1

Bl 27

Lista de componentes

F01_003BS

Número de artículo	Cantidad	Descripción	Código	Fabricante	BMK	Posición
PXC.3211766	6	Borne de tierra para carril	PT 4-PE	PXC	-X1	=M+S/1.8
PXC.3209536	4	Borne de tierra para carril	PT 2,5-PE	PXC	-X1	=M+S/1.8
PXC.3030417	1	Tapa final	D-ST 2,5	PHOENIX	-X1	=M+S/1.8
PXC.3030336	1	Puente enchufable	FBS 2-6	PXC	-X1	=M+S/1.8
PXC.3022276	2	Soporte final de montaje rápido	CLIPFIX 35-5	PXC	-X1	=M+S/1.8
PXC.3212066	1	Borne de tierra para carril	PTPOWER 35-PE	PXC	-X1	=M+S/1.8
PXC.3208155	1	Borne de paso	PT 1,5/S-TWIN	PXC	-X2	=M+S/2.8
PXC.3208168	1	Borne de paso	PT 1,5/S-TWIN BU	PXC	-X2	=M+S/2.8
PXC.3208184	1	Tapa final	D-PT 1,5/S-TWIN	PXC	-X2	=M+S/2.8
PXC.3208171	1	Borne de tierra para carril	PT 1,5/S-TWIN-PE	PXC	-X2	=M+S/2.8
PXC.3022276	2	Soporte final de montaje rápido	CLIPFIX 35-5	PXC	-X2	=M+S/2.8
PXC.3210600	18	Borne de doble piso	PTTBS 2,5-TWIN	PXC	-X3	=M+S/5.a.8
PXC.3210608	4	Tapa final	D-PTTBS 2,5-TWIN	PXC	-X3	=M+S/5.a.8
PXC.3030190	4	Puente enchufable	FBS 5-5	PXC	-X3	=M+S/5.a.8
PXC.3030187	4	Puente enchufable	FBS 4-5	PXC	-X3	=M+S/5.a.8
PXC.3022276	2	Soporte final	CLIPFIX 35-5	PXC	-X3	=M+S/5.a.8
PXC.3208155	2	Borne de paso	PT 1,5/S-TWIN	PXC	-X4	=M+S/6.9
PXC.3208168	2	Borne de paso	PT 1,5/S-TWIN BU	PXC	-X4	=M+S/6.9
PXC.3208184	1	Tapa final	D-PT 1,5/S-TWIN	PXC	-X4	=M+S/6.9
PXC.3208171	2	Borne de tierra para carril	PT 1,5/S-TWIN-PE	PXC	-X4	=M+S/6.9
PXC.3022276	2	Soporte final	CLIPFIX 35-5	PXC	-X4	=M+S/6.9
PXC.3213014	2	Puente enchufable	FBS 2-3,5	PXC	-X4	=M+S/6.9
PXC.3208155	12	Borne de tierra para carril	PT 1,5/S-TWIN	PXC	-X7	=M+S/7.8
PXC.3208184	2	Tapa final	D-PT 1,5/S-TWIN	PXC	-X7	=M+S/7.8
PXC.3208171	2	Borne de tierra para carril	PT 1,5/S-TWIN-PE	PXC	-X7	=M+S/7.8
PXC.3022276	2	Soporte final	CLIPFIX 35-5	PXC	-X7	=M+S/7.8
PXC.3208155	4	Borne de paso	PT 1,5/S-TWIN	PXC	-X10	=M+S/10.8
PXC.3208184	1	Tapa final	D-PT 1,5/S-TWIN	PXC	-X10	=M+S/10.8
PXC.3208171	1	Borne de tierra para carril	PT 1,5/S-TWIN-PE	PXC	-X10	=M+S/10.8
PXC.3022276	2	Soporte final	CLIPFIX 35-5	PXC	-X10	=M+S/10.8
PHO.3040012	2	Borne de paso	ST 2,5/ 1P	PHOENIX	-X39	=M+S/39.8
PXC.3040656	2	Borne de paso	ST 2,5/ 1P BU	PXC	-X39	=M+S/39.8
PXC.3040025	2	Borne de tierra de conexión por resorte	ST 2,5/ 1P-PE	PXC	-X39	=M+S/39.8
PXC.3210787	2	Conector	SP-H 2,5/ 1-L	PXC	-X39	=M+S/39.8
PXC.3210826	2	Conector	SP-H 2,5/ 1-M BU	PXC	-X39	=M+S/39.8
PXC.3210868	2	Conector	SP-H 2,5/ 1-R GNYE	PXC	-X39	=M+S/39.8
PXC.3030417	1	Tapa final	D-ST 2,5	PHOENIX	-X39	=M+S/39.8
PXC.3022276	2	Soporte final	CLIPFIX 35-5	PXC	-X39	=M+S/39.8

145

1

1.b

			Datum		...		Lista de componentes: PXC.3211766 -PXC.3022276	Besel & Schwäller Automationstechnik GmbH	= ST
			Bearb		Teststand Schraubmodul	BESEL & SCHWÄLLER Schaltanlagenbau GmbH			+
			Gepr						
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch		S...	Blatt 1.a
									Bl 27

Lista de componentes

F01_003BS

Artikelnummer	Menge	Bezeichnung	Typnummer	Hersteller	BMK	Schaltplan / Position
PXC.3244465	8	Borne para detectores/actuadores	PTIO 1,5/S/4-PE	PXC	-X40	=M+S/40.8
PXC.3244588	1	Tapa final	D-PTIO 1,5/S/4	PXC	-X40	=M+S/40.8
PXC.3244481	1	Borne para detectores/actuadores	PTIO-IN 2,5/4-PE OG	PXC	-X40	=M+S/40.8
PXC.3025341	2	Caballete	AB-SK	PXC	-X40	=M+S/40.8
PXC.3022276	2	Soporte final	CLIPFIX 35-5	PXC	-X40	=M+S/40.8
PXC.3213056	2	Puente enchufable	FBS 10-3,5	PXC	-X40	=M+S/40.8
PXC.3244465	8	Borne para detectores/actuadores	PTIO 1,5/S/4-PE	PXC	-X41	=M+S/41.9
PXC.3244588	1	Tapa final	D-PTIO 1,5/S/4	PXC	-X41	=M+S/41.9
PXC.3244481	1	Borne para detectores/actuadores	PTIO-IN 2,5/4-PE OG	PXC	-X41	=M+S/41.9
PXC.3025341	2	Caballete	AB-SK	PXC	-X41	=M+S/41.9
PXC.3022276	2	Soporte final	CLIPFIX 35-5	PXC	-X41	=M+S/41.9
PXC.3213056	2	Puente enchufable	FBS 10-3,5	PXC	-X41	=M+S/41.9
PXC.3244465	8	Borne para detectores/actuadores	PTIO 1,5/S/4-PE	PXC	-X42	=M+S/42.8
PXC.3244588	1	Tapa final	D-PTIO 1,5/S/4	PXC	-X42	=M+S/42.8
PXC.3244481	1	Bornes para detectores/actuadores	PTIO-IN 2,5/4-PE OG	PXC	-X42	=M+S/42.8
PXC.3022276	2	Soporte final	CLIPFIX 35-5	PXC	-X42	=M+S/42.8
PXC.3213056	4	Puente enchufable	FBS 10-3,5	PXC	-X42	=M+S/42.8
PXC.3046362	1	Borne interrumpible por cuchilla	UT 2,5-MT	PXC	-3X2	=M+S/3.2
PXC.3047141	1	Tapa final	D-UT 2,5/4-TWIN	PXC	-3X2	=M+S/3.2
PXC.3022276	2	Soporte final	CLIPFIX 35-5	PXC	-3X2	=M+S/3.2
PXC.3208100	2	Borne de paso	PT 1,5/S	PXC	-17X3	=M+S/17.8
PXC.3208142	1	Tapa final	D-PT 1,5/S	PXC	-17X3	=M+S/17.8
PXC.3022276	2	Soporte final	CLIPFIX 35-5	PXC	-17X3	=M+S/17.8

1.a

			Datum		...	BESEL & SCHWÄLLER Schaltanlagenbau GmbH	Lista de componentes: PXC.3244465 -PXC.3022276
			Bearb		Teststand Schraubmodul		
			Gepr				
Änderung	Datum	Name	Urspr		Ersatz von	Ersetzt durch	

Besel & Schwäller Automationstechnik GmbH

= ST

+

S...

Blatt 1.b

Bl 27

Apéndice E
Código

PROGRAM_INIT

```
iCounter                    :=0;

cKonfigAnalogIn.nRawMin          :=-32767;
cKonfigAnalogIn.nRawMax          :=32767;
cKonfigAnalogIn.nScaledMin       :=-10;
cKonfigAnalogIn.nScaledMax       :=10;
cKonfigAnalogIn.rFactor          :=1;
cKonfigAnalogIn.rOffset          :=0;

gBasicControlA01.Parameter.Acceleration          := 36000;  /*acceleration for movement*/
gBasicControlA01.Parameter.Deceleration          := 36000;  /*deceleration for movement*/
gBasicControlA01.Parameter.Direction                    := 1;
gBasicControlA01.Parameter.HomePosition                 := 0.0;
gBasicControlA01.Parameter.HomeMode                           := mcHOME_DIRECT;

/*******************Vorgegebene Parameter Array*******************/
asSpanner[0]     :=       "LC_90";
asSpanner[1]     :=       "LC_125";
asSpanner[2]     :=       "LC_160";
asSpanner[3]     :=       "LC_200";
asSpanner[4]     :=       "TC_90";
asSpanner[5]     :=       "TC_125";
asSpanner[6]     :=       "TC_160";
asSpanner[7]     :=       "TC_200";
asSpanner[8]     :=       "DUO_90";
asSpanner[9]     :=       "T-REX";
```

```
asSpanner[10]    :=        "TITAN_2_K";
asSpanner[11]    :=        "TITAN_2_M";
asSpanner[12]    :=        "TITAN_2_L";
asSpanner[13]    :=        "DUO_Plus_125";
asSpanner[14]    :=        "CENTRO_Gripp";
asSpanner[15]    :=        "CENTRO_Lite";
asSpanner[16]    :=        "Telecentric";

abKV[0]                    := TRUE;
abKV[1]                    := TRUE;
abKV[2]                    := TRUE;
abKV[3]                    := TRUE;
abKV[4]                    := TRUE;
abKV[5]                    := TRUE;
abKV[6]                    := TRUE;
abKV[7]                    := TRUE;
abKV[8]                    := TRUE;
abKV[9]                    := TRUE;
abKV[10]                   := TRUE;
abKV[11]                   := TRUE;
abKV[12]                   := TRUE;
abKV[13]                   := TRUE;
abKV[14]                   := TRUE;
abKV[15]                   := FALSE;
abKV[16]                   := FALSE;

arMKraft[0]                := 34;
arMKraft[1]                := 48;
arMKraft[2]                := 72;
arMKraft[3]                := 72;
arMKraft[4]                := 34;
arMKraft[5]                := 48;
```

```
arMKraft[6]             := 72;
arMKraft[7]             := 96;
arMKraft[8]             := 34;
arMKraft[9]             := 48;
arMKraft[10]    := 48;
arMKraft[11]    := 48;
arMKraft[12]    := 48;
arMKraft[13]    := 55;
arMKraft[14]    := 48;
arMKraft[15]    := 48;
arMKraft[16]    := 72;

arMMoment[0]    := 30;
arMMoment[1]    := 45;
arMMoment[2]    := 55;
arMMoment[3]    := 55;
arMMoment[4]    := 30;
arMMoment[5]    := 45;
arMMoment[6]    := 55;
arMMoment[7]    := 55;
arMMoment[8]    := 35;
arMMoment[9]    := 35;
arMMoment[10]   := 35;
arMMoment[11]   := 35;
arMMoment[12]   := 35;
arMMoment[13]   := 45;
arMMoment[14]   := 75;
arMMoment[15]   := 115;
arMMoment[16]   := 55;

arSteigung[0]   := 5;
arSteigung[1]   := 6;
```

```
arSteigung[2]    := 6;
arSteigung[3]    := 6;
arSteigung[4]    := 5;
arSteigung[5]    := 6;
arSteigung[6]    := 6;
arSteigung[7]    := 6;
arSteigung[8]    := 5;
arSteigung[9]    := 5;
arSteigung[10]   := 6;
arSteigung[11]   := 6;
arSteigung[12]   := 6;
arSteigung[13]   := 5;
arSteigung[14]   := 5;
arSteigung[15]   := 1.5;
arSteigung[16]   := 2;

/*****************************************************************/

/******************Vorgegebene Parameter**************************/

uHub                    := 10;

rSteigung               := 6;

rOffnungsWeite   := 3;

uMod                    := 1;

tDel1                   := 10;
tDel2                   := 10;

rMaxKraft               := 20;
```

```
rMaxMoment                  := 15;
iWinkelPosReg    := 30;

IF uKV = 1 THEN

rGKraft                     := 150;

ELSE

rGKraft                     := 40;

END_IF

bKV                                 := TRUE;

uKV                                 := 1;

rGMoment                    := 150;

/*****************************************************************/

END_PROGRAM


PROGRAM _CYCLIC


/*****************************************************************
*****                    Filtrieren                          ******
*****************************************************************/

rKraftFiltriert40kN                 := (rKraftIst40kN + k2xn1 + k2xn2 + k2xn3 + k2xn4 + k2xn5 + k2xn6
```

```
   + k2xn7 + k2xn8 + k2xn9 + k2xn10)/(11);


/*****************************************************************
*****              Kraft (40kN) Berechnen                    ******
*****************************************************************/

BuR_ScaleAnalogIn(iKraftSensor1 ,cKonfigAnalogIn ,rKraftIst40kN);

rKraft40kN:=Kraft40(rKraftIst40kN);

/*****************************************************************
*****                    Filtrieren                          ******
*****************************************************************/

rKraftFiltriert150kN     := (rKraftIst150kN + kxn1 + kxn2 + kxn3 + kxn4 + kxn5 + kxn6 + kxn7 + kxn8
    + kxn9 + kxn10)/(11);

/*****************************************************************
*****              Kraft (150kN) Berechnung                  ******
*****************************************************************/

BuR_ScaleAnalogIn(iKraftSensor2 ,cKonfigAnalogIn ,rKraftIst150kN);

rKraft150kN:=Kraft150(rKraftFiltriert150kN);

/*****************************************************************
*****              Drehmoment Berechnung                     ******
*****************************************************************/

BuR_ScaleAnalogIn(iMomentSensor ,cKonfigAnalogIn ,rMomentIst);
rMn1     := rMomentIst*20;
```

```
rMoment          := (rMn1 + mxn1 + mxn2 + mxn3 + mxn4 + mxn5 + mxn6 + mxn7 + mxn8 + mxn9 + mxn10)
    /11;

/*****************************************************************
*****              Drehmoment Ableitung                      ******
*****************************************************************/

IF (gBasicControlA01.Status.ActPosition-rPosn1 = 0) THEN
rDerMp := 0;
ELSE
rDerMp := 10000*(rMoment-rDn)/ABS(gBasicControlA01.Status.ActPosition-rPosn1);
END_IF

/*****************************************************************
*****          Drehmoment Ableitung filtrieren               ******
*****************************************************************/

rDerM            := (rDerMp + rDerMn1 + rDerMn2 + rDerMn3 + rDerMn4 + rDerMn5 + rDerMn6 + rDerMn7 +
    rDerMn8 + rDerMn9 + rDerMn10)/11;

/*****************************************************************
*****     Werte speichern für Ableitung Berechnung           ******
*****************************************************************/

rDn                      := rMoment;
rPosn1           := gBasicControlA01.Status.ActPosition;

/*****************************************************************
*****************************************************************
Die Spanner mit Anschlag können auch mit den anderen Steuerungsarten gesteuert werden, aber die
   vordefinierte Art ist mit Winkel
```

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

IF bAnschlag THEN

uMod      := 2;

END_IF

/*************************************************************************
*****                  Berechnete Werte                          *******
*************************************************************************/

rDerMKup          := 540;
rDerMKup2         := -380;
rDerMKupNeg       := -500;

/*************************************************************************
*****                          Delays                             ******
*************************************************************************/

tPZ1      := UDINT_TO_TIME(tDel1*1000);
tPZ2      := UDINT_TO_TIME(tDel2*1000);

tDelay(IN:= (uStateMachine=1010 OR uStateMachine = 2020 OR uStateMachine = 3010),PT:=tPZ1);
tDelayRW(IN:=(uStateMachine=1000 OR uStateMachine=2000 OR uStateMachine=3000),PT:=tPZ2);

/*************************************************************************
*****              Parameter aus Liste lesen                     ******
*****  nur wenn die Parameter mode "vordefinierte" ist           ******
*************************************************************************/
```

```
IF NOT bParMode THEN

sSpanner          := asSpanner[uAuswahl];
bKV                     := abKV[uAuswahl];
rGKraft           := arMKraft[uAuswahl];
rGMoment          := arMMoment[uAuswahl];
rSteigung         := arSteigung[uAuswahl];

ELSE

sSpanner          := "";

END_IF

/*****************************************************************
*****                    System Fehler                      ******
*****************************************************************/

bErrorState       := FALSE;
sErrorText        := "";

IF bErrorStateErk THEN

bAutoBetrieb      := FALSE;

END_IF

IF (rKraft150kN >= 5) AND (rKraft40kN >= 5) THEN

bErrorState       := TRUE;
sErrorText        := "Kraftmessdosen Kontrolle erforderlich";
```

```
END_IF

IF (rMoment >= rGMoment/1.5) AND ((rKraft150kN <= 2) AND (rKraft40kN <= 2)) THEN

bErrorState      := TRUE;
sErrorText       := "Kraftmessdose nicht/falsch montiert/erkannt/kalibriert oder Spinndel verstopft
   ";

END_IF

IF uKV = 1 THEN

IF (rKraft150kN > rGKraft) THEN

bErrorState      := TRUE;
sErrorText       := "Maximaler Kraft Wert des ausgewählten Spanner wurden überschritten";

END_IF

ELSE

IF (rKraft40kN > rGKraft) THEN

bErrorState      := TRUE;
sErrorText       := "Maximaler Kraft Wert des ausgewählten Spanner wurden überschritten";

END_IF

END_IF

IF (rMoment > rGMoment) AND ((uStateMachine > 52) OR (uStateMachine < 20)) THEN
```

```
bErrorState      := TRUE;
sErrorText       := "Maximaler Drehmoment Wert des ausgewählten Spanner wurden überschritten";

END_IF

IF (rMoment > 1.5*rGMoment) AND ((uStateMachine <= 52) OR (uStateMachine >= 20)) THEN

bErrorState      := TRUE;
sErrorText       := "Maximaler Drehmoment Wert des ausgewählten Spanner wurden überschritten";

END_IF

IF gBasicControlA01.Status.ErrorID <> 0 THEN

bErrorState      := TRUE;
sErrorText       := "Achse Fehler";

END_IF

IF ((rKraft150kN > 150) OR (rMoment > 150) OR (rKraft40kN >= 40)) THEN

bErrorState      := TRUE;
sErrorText       := "Maximale allgemeine Last Werte wurden überschritten";

END_IF

/*****************************************************************
*****                  Fehler Stop                          ******
*****************************************************************/

IF bErrorState AND (NOT bErrorStateErk) THEN
```

```
gBasicControlA01.Command.Power                    := FALSE;
gBasicControlA01.Command.Home                     := FALSE;
gBasicControlA01.Command.MoveAbsolute          := FALSE;
gBasicControlA01.Command.MoveAdditive          := FALSE;
gBasicControlA01.Command.MoveVelocity          := FALSE;
gBasicControlA01.Command.Halt                     := FALSE;
gBasicControlA01.Command.Stop                     := FALSE;
gBasicControlA01.Command.MoveJogPos               := FALSE;
gBasicControlA01.Command.MoveJogNeg               := FALSE;
bStepEnable                                                    := FALSE;
bPowerOn                                                       := FALSE;
bHome                                                          := FALSE;
uStateMachine                                                  := 0;
rStartPosition                                                 := 0;
rEndPosition                                                   := 0;
bAutoBetrieb                                                   := FALSE;

END_IF

/*****************************************************************
******            Position Grenze Fehler                  *******
*****************************************************************/

IF NOT bInduktiv AND NOT bIndBypass THEN

uStateMachine    := 5000;

END_IF

IF bInduktiv AND bIndBypass THEN

bIndBypass               := FALSE;
```

```
END_IF

/*****************************************************************
*******        Winkel nach Einheiten umrechnen              ********
*****************************************************************/


rDistance        := iWinkelPosReg*10;
rStartPos        := rStartPosition/10;

/*****************************************************************
*****       Fehler beim Datei Speichern Handeln                 ******
*****************************************************************/

IF bFileError THEN

bStepEnable      := FALSE;

END_IF

/*****************************************************************
************            Error Reset                 ***************
*****************************************************************/

IF gBasicControlA01.Status.DriveStatus.AxisError AND gBasicControlA01.Command.ErrorAcknowledge
   THEN

gBasicControlA01.Command.ErrorAcknowledge        := FALSE;

END_IF
```

```
IF (bErrorReset) AND (gBasicControlA01.Status.ErrorID <> 0) THEN

gBasicControlA01.Command.ErrorAcknowledge       := TRUE;
uStateMachine                                                   := 0;
bErrorReset                                                             := FALSE;


END_IF


IF  gBasicControlA01.Status.ErrorID = 0 THEN

bErrorReset                                                             := FALSE;
gBasicControlA01.Command.ErrorAcknowledge       := FALSE;

END_IF

(*****************************************************************
*****************************************************************
*****************************************************************
*****
                        ******
*****                   STATE MACHINE ALGORITHMUS                 ******
*****                                                             ******
*****************************************************************
*****************************************************************
*****************************************************************)

CASE uStateMachine OF

0:      (*****************     Antrieb Einschalten             *****************)
```

```
IF bStepEnable AND bPowerOn THEN

gBasicControlA01.Command.Power          := TRUE;
uStateMachine                                   := 5;
iCounter:=0;

END_IF

5:      /*****************      Referenzieren (Homing)      ******************/

IF bStepEnable AND bHome THEN

gBasicControlA01.Command.Home           := TRUE;
bStepEnable                                             := FALSE;
uStateMachine                                   := 10;

END_IF

10:     /*****************   Referenzieren kontrollieren   ******************/

IF gBasicControlA01.AxisState.HomeDone THEN

gBasicControlA01.Command.Home           := FALSE;
bHome                                                   := FALSE;
gBasicControlA01.AxisState.HomeDone := FALSE;
uStateMachine                                   := 20;

END_IF

20:

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

```
************      Auto/Manueller Betrieb                 ****************
*****************************************************************/

gBasicControlA01.Command.Halt    := FALSE;

IF (bAutoBetrieb) AND (rStartPosition <> 0) AND (rEndPosition <> 0) THEN

uStateMachine                                             := 30;

ELSE

uStateMachine                                             := 40;

END_IF

/*****************************************************************
************            Referenzieren                  ****************
*****************************************************************/

IF bHome THEN

uStateMachine                                             := 5;
gBasicControlA01.Command.MoveJogNeg       := FALSE;
gBasicControlA01.Command.MoveJogPos       := FALSE;
rStartPosition                                            := 0;
rEndPosition                                              := 0;

END_IF

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

```
************            Ein/ausschalten              *****************
*****************************************************************/

IF NOT bPowerOn THEN

uStateMachine                                                := 0;
gBasicControlA01.Command.MoveJogNeg       := FALSE;
gBasicControlA01.Command.MoveJogPos       := FALSE;
gBasicControlA01.Command.Power            := FALSE;

END_IF

30:      /*****************  Steuerungsart wählen            ******************/

bStepEnable      := FALSE;

CASE uMod OF

1:

bWerteUngueltig := TRUE;

IF (rMaxKraft >= 10.0) THEN
/* Parameter kontrollieren */

uStateMachine    := 1000;
bWerteUngueltig :=FALSE;

ELSE

bAutoBetrieb     := FALSE;
```

```
END_IF

2:
bWerteUngueltig := TRUE;

IF (rDistance > 0.0) AND (rDistance < rEndPosition-rStartPosition) THEN
/* Parameter kontrollieren */

uStateMachine   := 2000;
bWerteUngueltig := FALSE;

ELSE

bAutoBetrieb    := FALSE;

END_IF

3:
bWerteUngueltig := TRUE;

IF (rMaxMoment >= 10) THEN
/* Parameter kontrollieren */

uStateMachine   := 3000;
bWerteUngueltig := FALSE;

ELSE

bAutoBetrieb    := FALSE;

END_IF
```

```
END_CASE

40:

/*****************************************************************
************          Auf/zumachen          ********************
*****************************************************************/

CASE bDirection OF

0:
uStateMachine    := 51;

1:
uStateMachine    := 52;

END_CASE

/*****************************************************************
************     Anfang/Ende Position speichern    ********************
*****************************************************************/

IF bStartPosSpeich THEN

IF gBasicControlA01.Status.ActPosition = 0 THEN

rStartPosition   := -1;

ELSE

rStartPosition   := gBasicControlA01.Status.ActPosition;
```

```
END_IF

bStartPosSpeich          := FALSE;

END_IF

IF bEndPosSpeich THEN

IF gBasicControlA01.Status.ActPosition = 0 THEN

rEndPosition     := 1;

ELSE

rEndPosition             := gBasicControlA01.Status.ActPosition;

END_IF

bEndPosSpeich            := FALSE;

END_IF

51: /******************        Aufmachen                    ********************/

IF bMove THEN

gBasicControlA01.Command.MoveJogPos :=FALSE;
gBasicControlA01.Command.MoveJogNeg      := bStepEnable;

/*****************************************************************
***********       Position Grenze Fehler            ***************
*****************************************************************/
```

```
IF bInduktiv THEN

bIndBypass                                                    := FALSE;

END_IF

uStateMachine                                                 := 20;

/*****************************************************************
***********              Ausrasten erkennen            ***************
*****************************************************************/


IF (rDerM<=-800) AND (rDerM>=-2500) AND (rKraft150kN <= 2) AND  (rKraft40kN <= 2) AND bKV THEN

rStartPosition := gBasicControlA01.Status.ActPosition-(rOffnungsWeite+1)/rSteigung*3600;

END_IF

ELSE

gBasicControlA01.Command.MoveJogNeg := FALSE;
uStateMachine                                                 := 20;

END_IF

52: /*****************          Zumachen                  ******************/

IF bMove THEN

gBasicControlA01.Command.MoveJogNeg :=FALSE;
```

```
gBasicControlA01.Command.MoveJogPos      := bStepEnable;

/*****************************************************************
************          Anschlag erkennen              ***************
*****************************************************************/

IF (rDerM > 7500) THEN

uMod                                                    := 2;
bMove                                                   := FALSE;
bStepEnable                                             := FALSE;
gBasicControlA01.Command.MoveJogPos      := FALSE;
bAnschlag                                               := TRUE;

END_IF

/*****************************************************************
************        Position Grenze Fehler           ***************
*****************************************************************/

IF bInduktiv THEN

bIndBypass                                              := FALSE;

END_IF

uStateMachine                                           := 20;

/*****************************************************************
************   Ende Position automatisch speichern ***************
*****************************************************************/
```

```
CASE uMod OF

1:
IF (rKraft150kN >= 1.1*rMaxKraft) OR (rKraft40kN >= 1.1*rMaxKraft) THEN

bEndPosSpeich    := TRUE;
bStepEnable              := FALSE;

END_IF

2:
IF rDerM > 7500 THEN

bEndPosSpeich    := TRUE;
bStepEnable              := FALSE;

END_IF

3:
bLimitM:= ((rMoment >= 1.1*rMaxMoment) AND       ((rKraft150kN >= 2) OR (rKraft40kN >= 2)));

IF bLimitM THEN

bEndPosSpeich    := TRUE;
bStepEnable              := FALSE;

END_IF

END_CASE


ELSE
```

```
gBasicControlA01.Command.MoveJogPos := FALSE;
uStateMachine                                := 20;

END_IF



/*****************************************************************
*****************************************************************
************          KRAFT REGELUNG                ***************
*****************************************************************
*****************************************************************/



1000: /*****************          Zumachen                ********************/

/*****************************************************************
************          Test abbrechen?                ***************
*****************************************************************/

IF bTestAbbrechen THEN

uStateMachine                                                := 20;
iCounter                                                            := 0;
bAutoBetrieb                                                 := FALSE;
bTestAbbrechen                                               := FALSE;
bStepEnable                                                         := FALSE;
gBasicControlA01.Command.Halt                    := TRUE;
gBasicControlA01.Command.MoveAbsolute    := FALSE;
```

```
END_IF

/*****************************************************************
***********          Neuer Test?                  *****************
*****************************************************************/

IF bNeueTest THEN

uStateMachine                                                := 20;
bNeueTest                                                          := FALSE;
iCounter                                                           := 0;
bStepEnable                                                        := FALSE;

END_IF

gBasicControlA01.Parameter.Position                  := rEndPosition;
gBasicControlA01.Command.Halt                        := FALSE;
gBasicControlA01.Command.MoveAbsolute    := bStepEnable AND (NOT bLimitK) AND tDelayRW.Q;

/*****************************************************************
***********          Werte speichern              *****************
*****************************************************************/

IF (rKraftHolder <> 0) THEN

rKraftErreicht  := rKraftHolder;

END_IF

IF (rLetzteVal = rMaxKraft)  AND (rKraftErreicht > 0) AND (rKraftKorr > 0) AND bKorr THEN
/*****************  Kraft Korrektur (Regelungsprozess) *****************/
```

```
rKraftKorr                  := rKraftKorr*(2+rKraftErreicht/rMaxKraft)/3;
bKorr                       := FALSE;

END_IF

IF rKraftKorr <= 1 THEN
/******************          Overshoot begrenzen          ********************/

rKraftKorr                  := 1;

END_IF

IF rMHolder      <> 0 THEN

rMomentErreicht := rMHolder;

END_IF

IF rWHolder <> 0 THEN

rWinkelErreicht := rWHolder/10;

END_IF

IF rK2MHolder <> 0 THEN

rK2MomentErreicht           := rK2MHolder;

END_IF


rKraftHolder                                                    := 0;     //Holder rücksetzen
```

```
rMHolder                                                    := 0;
rWHolder                                                    := 0;
rK2MHolder                                                  := 0;

IF (rDerM > 0.5*rDerMKup) AND (rMoment > 0.95*rKMomentErreicht) AND (rMoment < myn1) THEN

rPosKup1                        := gBasicControlA01.Status.ActPosition/10;

IF rMoment > rKMHolder THEN

rKMHolder                       := rMoment;

END_IF

END_IF

/*****************************************************************
***********        Zumachen Ende Bedingung          ***************
*****************************************************************/

bLimitK := ((rKraft150kN >= rMaxKraft/rKraftKorr) OR (rKraft40kN >= rMaxKraft/rKraftKorr));

IF bLimitK THEN

gBasicControlA01.Command.Halt           := TRUE;
uStateMachine                                       := 1010;

END_IF

/*****************************************************************
***********              Fehler                     ***************
*****************************************************************/
```

```
IF (rDerM > 7500) THEN

gBasicControlA01.Command.Halt          := TRUE;
uStateMachine                                  := 0;
bStepEnable                                           := FALSE;
bErrorState                                           := TRUE;
bAnschlag                                             := TRUE;
bAutoBetrieb                                   := FALSE;

END_IF


IF gBasicControlA01.AxisState.MoveAbsoluteDone THEN

uStateMachine                                  := 1040;

END_IF

1010:   /*****************        Aufmachen                  ********************/

/*****************************************************************
************        Test abbrechen?              ****************
*****************************************************************/

IF bTestAbbrechen THEN

uStateMachine                                          := 20;
iCounter                                                      := 0;
bAutoBetrieb                                           := FALSE;
bTestAbbrechen                                         := FALSE;
bStepEnable                                                   := FALSE;
```

```
gBasicControlA01.Command.Halt                    := TRUE;
gBasicControlA01.Command.MoveAbsolute    := FALSE;

END_IF

gBasicControlA01.Parameter.Position              := rStartPosition;
gBasicControlA01.Command.Halt                    := FALSE;
gBasicControlA01.Command.MoveAbsolute    := bStepEnable AND tDelay.Q;
bKorr                                                                    := TRUE;

/*****************************************************************
***********          Werte vorbereiten              ***************
*****************************************************************/

IF rKraft150kN > rKraftHolder THEN

rKraftHolder     := rKraft150kN;

END_IF

IF rKraft40kN > rKraftHolder THEN

rKraftHolder     := rKraft40kN;

END_IF

IF rMoment > rMHolder THEN

rMHolder                 := rMoment;

END_IF
```

```
IF gBasicControlA01.Status.ActPosition > rWHolder THEN
rWHolder                    := gBasicControlA01.Status.ActPosition;
END_IF
IF rKMHolder <> 0 THEN
rKMomentErreicht            := rKMHolder;
END_IF
IF rMoment < rK2MHolder THEN
rK2MHolder                          := rMoment;
END_IF

rKMHolder                           := 0; //Holder rücksetzen

/*****************************************************************
***********          Position Speichern              ***************
*****************************************************************/

IF (rDerM < 0.5*rDerMKupNeg) AND (rDerM> -1000)AND (rMoment < 0.5*rK2MomentErreicht) THEN
rPosKup2                    := gBasicControlA01.Status.ActPosition/10;
END_IF
```

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

IF gBasicControlA01.AxisState.MoveAbsoluteDone THEN

gBasicControlA01.Command.MoveAbsolute :=FALSE;

/*****************************************************************
************          Hübe zählen                  ****************
*****************************************************************/

IF iCounter <= uHub THEN

iCounter                := iCounter + 1;
uStateMachine    := 1000;

ELSE

/*****************************************************************
************          Test beenden                 ****************
*****************************************************************/

uStateMachine    := 1030;
iCounter                := 0;
bStepEnable             := FALSE;

END_IF

END_IF

1030:    /*****************          Test fertig          ******************/
```

```
/*****************************************************************
************     Zum Manuellen Betrieb                ***************
*****************************************************************/

IF NOT bAutoBetrieb THEN

uStateMachine    := 20;
bMove                    := FALSE;

END_IF

/*****************************************************************
************      Neuer (Kraft) Test                  ***************
*****************************************************************/

IF bNeueTest THEN

uStateMachine    := 20;
bNeueTest                := FALSE;
iCounter                 := 0;
bStepEnable              := FALSE;

END_IF

1040:    /*****************  Zum Anfang Position nach Fehler  ******************/

gBasicControlA01.Parameter.Position                    := rStartPosition;
gBasicControlA01.Command.Halt                          := FALSE;
gBasicControlA01.Command.MoveAbsolute          := TRUE;

IF gBasicControlA01.AxisState.MoveAbsoluteDone THEN
```

```
gBasicControlA01.Command.MoveAbsolute          := FALSE;
gBasicControlA01.AxisState.MoveAbsoluteDone    := FALSE;
uStateMachine                                                  := 1050;

END_IF

1050: /*****************     Fehler Zustand                  ********************/

/*****************************************************************
************          Zum Manuellen Betrieb          ****************
*****************************************************************/

IF bTestAbbrechen THEN

uStateMachine    := 20;
iCounter                 := 0;
bAutoBetrieb     := FALSE;
bTestAbbrechen   := FALSE;
bStepEnable              := FALSE;

END_IF

/*****************************************************************
************          Ende Position ändern           ****************
*****************************************************************/

IF bErrorEnd THEN

bStepEnable      := FALSE;
rEndPosition     := rEndPosition + iWinkelKorr*10;
uStateMachine    := 1000;
bErrorEnd                := FALSE;
```

```
END_IF



/*****************************************************************
*****************************************************************
************          WINKEL REGELUNG              *****************
*****************************************************************
*****************************************************************/



2000: /******************          Zumachen                ********************/

/*****************************************************************
************          Test abbrechen?              *****************
*****************************************************************/

IF bTestAbbrechen THEN

uStateMachine                                                   := 20;
iCounter                                                                := 0;
bAutoBetrieb                                                    := FALSE;
bTestAbbrechen                                                  := FALSE;
bStepEnable                                                             := FALSE;
gBasicControlA01.Command.Halt                           := TRUE;
gBasicControlA01.Command.MoveAbsolute     := FALSE;

END_IF

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

```
************          Neuer Test?                    ******************
*****************************************************************/

IF bNeueTest THEN

uStateMachine    := 20;
bNeueTest                := FALSE;
iCounter                 := 0;
bStepEnable              := FALSE;

END_IF

gBasicControlA01.Parameter.Position              := rEndPosition - rDistance;
gBasicControlA01.Command.Halt                    := FALSE;
gBasicControlA01.Command.MoveAbsolute    := bStepEnable AND tDelayRW.Q;

/*****************************************************************
************              Fehler                     ******************
*****************************************************************/

IF rDerM >= 7500 THEN

uStateMachine    := 2040;

END_IF

/*****************************************************************
************          Werte speichern                ******************
*****************************************************************/

IF rKraftHolder <> 0 THEN
```

```
rKraftErreicht   := rKraftHolder;

END_IF

IF rMHolder      <> 0 THEN

rMomentErreicht := rMHolder;

END_IF

IF rWHolder <> 0 THEN

rWinkelErreicht := rWHolder/10;

END_IF


IF rK2MHolder <> 0 THEN

rK2MomentErreicht        := rK2MHolder;

END_IF

rKraftHolder                                                    := 0;     //Holder Rücksetzen
rMHolder                                                                  := 0;
rWHolder                                                                  := 0;
rK2MHolder                                                                := 0;


IF (rDerM > 0.5*rDerMKup) AND (rMoment > 0.95*rKMomentErreicht) THEN

rPosKup1                 := gBasicControlA01.Status.ActPosition/10;
```

```
IF rMoment > rKMHolder THEN

rKMHolder                   := rMoment;

END_IF

END_IF

/*****************************************************************
***********        Zumachen Ende Bedingung          ***************
*****************************************************************/

IF gBasicControlA01.AxisState.MoveAbsoluteDone THEN

gBasicControlA01.Command.MoveAbsolute             :=FALSE;
gBasicControlA01.AxisState.MoveAbsoluteDone := FALSE;
uStateMachine := 2010;

END_IF

2010:

IF NOT gBasicControlA01.AxisState.MoveAbsoluteDone THEN

uStateMachine := 2020;

END_IF


2020:   /*****************          Aufmachen                  ******************/
```

```
/*****************************************************************
************          Test abbrechen?              ****************
*****************************************************************/

IF bTestAbbrechen THEN

uStateMachine                                                := 20;
iCounter                                                           := 0;
bAutoBetrieb                                                 := FALSE;
bTestAbbrechen                                               := FALSE;
bStepEnable                                                        := FALSE;
gBasicControlA01.Command.Halt                        := TRUE;
gBasicControlA01.Command.MoveAbsolute    := FALSE;

END_IF

gBasicControlA01.Parameter.Position                  := rStartPosition;
gBasicControlA01.Command.MoveAbsolute    := bStepEnable AND tDelay.Q;

/*****************************************************************
************          Werte vorbereiten            ****************
*****************************************************************/

IF rKraft150kN > rKraftHolder THEN

rKraftHolder     := rKraft150kN;

END_IF

IF rKraft40kN > rKraftHolder THEN

rKraftHolder     := rKraft40kN;
```

```
END_IF

IF rMoment > rMHolder THEN

rMHolder                    := rMoment;

END_IF

IF gBasicControlA01.Status.ActPosition > rWHolder THEN

rWHolder                    := gBasicControlA01.Status.ActPosition;

END_IF

IF rKMHolder <> 0 THEN

rKMomentErreicht:= rKMHolder;

END_IF

IF (rMoment < rK2MHolder) AND (rDerM < 0.5*rDerMKupNeg) THEN

rK2MHolder                  := rMoment;

END_IF


rKMHolder                   := 0; //Hodler rücksetzen

/*****************************************************************
***********          Position Speichern              ***************
```

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

IF (rDerM < 0.5*rDerMKupNeg) AND (rDerM> -1000)AND (rMoment < 0.95*rK2MomentErreicht) AND (rMoment
    > myn1) THEN

rPosKup2                    := gBasicControlA01.Status.ActPosition/10;

END_IF

/*************************************************************/

IF gBasicControlA01.AxisState.MoveAbsoluteDone THEN

gBasicControlA01.Command.MoveAbsolute :=FALSE;

/*****************************************************************
************          Hübe zählen                  ****************
*****************************************************************/

IF iCounter <= uHub THEN

iCounter                    := iCounter + 1;
uStateMachine    := 2000;

ELSE

/*****************************************************************
************          Test beenden                 ****************
*****************************************************************/

uStateMachine    := 2030;
iCounter                    := 0;
```

```
bStepEnable             := FALSE;

END_IF

END_IF

2030:   /*****************          Test fertig             ********************/

/*****************************************************************
************    Zum Manuellen Betrieb              ****************
*****************************************************************/

IF NOT bAutoBetrieb THEN

uStateMachine   := 20;
bMove                   := FALSE;

END_IF

/*****************************************************************
************     Neuer (Winkel) Test                ****************
*****************************************************************/

IF bNeueTest THEN

uStateMachine   := 20;
bNeueTest               := FALSE;
iCounter                := 0;
bStepEnable             := FALSE;

END_IF
```

```
2040: /******************  Zum Anfang Position nach Fehler  ******************/

gBasicControlA01.Parameter.Position                    := rStartPosition;
gBasicControlA01.Command.Halt                          := FALSE;
gBasicControlA01.Command.MoveAbsolute          := TRUE;

IF gBasicControlA01.AxisState.MoveAbsoluteDone THEN

gBasicControlA01.Command.MoveAbsolute          := FALSE;
gBasicControlA01.AxisState.MoveAbsoluteDone     := FALSE;
uStateMachine                                                  := 2050;

END_IF

2050: /******************     Fehler Zustand              ********************/

/*****************************************************************
************         Zum Manuellen Betrieb          ****************
*****************************************************************/

IF bTestAbbrechen THEN

uStateMachine    := 20;
iCounter                 := 0;
bAutoBetrieb     := FALSE;
bTestAbbrechen   := FALSE;
bStepEnable              := FALSE;

END_IF
```

```
/*****************************************************************
*****************************************************************
************          DREHMOMENT REGELUNG                 ***********
*****************************************************************
*****************************************************************/


3000: /*******************          Zumachen                 ********************/

/*****************************************************************
************          Test abbrechen?                 ***********
*****************************************************************/

IF bTestAbbrechen THEN

uStateMachine                                                   := 20;
iCounter                                                                := 0;
bAutoBetrieb                                                    := FALSE;
bTestAbbrechen                                                  := FALSE;
bStepEnable                                                             := FALSE;
gBasicControlA01.Command.Halt                           := TRUE;
gBasicControlA01.Command.MoveAbsolute     := FALSE;

END_IF

/*****************************************************************
************          Neuer Test?                 ***********
*****************************************************************/

IF bNeueTest THEN
uStateMachine                                                   := 20;
```

```
bNeueTest                                                                    := FALSE;
iCounter                                                                     := 0;
bStepEnable                                                                  := FALSE;
END_IF

gBasicControlA01.Parameter.Position                  := rEndPosition;
gBasicControlA01.Command.Halt                        := FALSE;
gBasicControlA01.Command.MoveAbsolute    := bStepEnable AND (NOT bLimitM) AND tDelayRW.Q;

/*****************************************************************
***********          Werte speichern                ***************
*****************************************************************/

IF (rKraftHolder <> 0) THEN

rKraftErreicht  := rKraftHolder;

END_IF

IF (rLetzteVal = rMaxMoment)  AND (rMomentErreicht > 0) AND (rMomKorr > 0) AND bKorr THEN
/************** Drehmoment Korrektur (Regelungsprozess) ***************/

rMomKorr                  := rMomKorr*(2+rMomentErreicht/rMaxMoment)/3;
bKorr                     := FALSE;

END_IF

IF rMomKorr <= 1 THEN
/*****************        Overshoot begrenzen        ******************/
rMomKorr                  := 1;

END_IF
```

```
IF rMHolder      <> 0 THEN

rMomentErreicht := rMHolder;

END_IF

IF rWHolder <> 0 THEN

rWinkelErreicht := rWHolder/10;

END_IF

IF rK2MHolder <> 0 THEN

rK2MomentErreicht        := rK2MHolder;

END_IF


rKraftHolder                                                    := 0;     //Holder Rücksetzen
rMHolder                                                              := 0;
rWHolder                                                              := 0;
rK2MHolder                                                            := 0;

IF (rDerM > 0.5*rDerMKup) AND (rMoment > 0.95*rKMomentErreicht) AND (rMoment < myn1) THEN

rPosKup1                 := gBasicControlA01.Status.ActPosition/10;

IF rMoment > rKMHolder THEN

rKMHolder                := rMoment;
```

```
END_IF

END_IF

/*****************************************************************
************        Zumachen Ende Bedingung          ***************
*****************************************************************/

bLimitM:= ((rMoment >= rMaxMoment/rMomKorr) AND ((rKraft150kN > 7.5) OR (rKraft40kN > 7.5 )));


IF bLimitM THEN

gBasicControlA01.Command.Halt                    := TRUE;
uStateMachine                                                := 3010;

END_IF

/*****************************************************************
************              Fehler                     ***************
*****************************************************************/

IF (rDerM > 7500) THEN

gBasicControlA01.Command.Halt          := TRUE;
uStateMachine                                        := 0;
bStepEnable                                                := FALSE;
bErrorState                                                := TRUE;
bAnschlag                                                  := TRUE;
bAutoBetrieb                                         := FALSE;
```

```
END_IF

IF gBasicControlA01.AxisState.MoveAbsoluteDone THEN

uStateMachine                                                    := 3040;

END_IF

3010: /******************          Aufmachen                  ********************/

/*****************************************************************
************          Test abbrechen?                 ****************
*****************************************************************/

IF bTestAbbrechen THEN

uStateMachine                                                    := 20;
iCounter                                                                 := 0;
bAutoBetrieb                                                     := FALSE;
bTestAbbrechen                                                   := FALSE;
bStepEnable                                                              := FALSE;
gBasicControlA01.Command.Halt                          := TRUE;
gBasicControlA01.Command.MoveAbsolute     := FALSE;

END_IF

gBasicControlA01.Parameter.Position                    := rStartPosition;
gBasicControlA01.Command.Halt                          := FALSE;
gBasicControlA01.Command.MoveAbsolute     := bStepEnable AND tDelay.Q;
bKorr                                                            := TRUE;

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

```
************            Werte vorbereiten              ****************
*****************************************************************/

IF rKraft150kN > rKraftHolder THEN

rKraftHolder     := rKraft150kN;

END_IF

IF rKraft40kN > rKraftHolder THEN

rKraftHolder     := rKraft40kN;

END_IF

IF rMoment > rMHolder THEN

rMHolder                 := rMoment;

END_IF

IF gBasicControlA01.Status.ActPosition > rWHolder THEN

rWHolder                 := gBasicControlA01.Status.ActPosition;

END_IF

IF rKMHolder <> 0 THEN

rKMomentErreicht:= rKMHolder;

END_IF
```

```
IF rMoment < rK2MHolder THEN

rK2MHolder                  := rMoment;

END_IF


rKMHolder                           := 0;


/*****************************************************************
***********          Position Speichern              ***************
*****************************************************************/


IF (rDerM < 0.5*rDerMKupNeg) AND (rDerM> -1000)AND (rMoment < 0.35*rK2MomentErreicht) THEN

rPosKup2                    := gBasicControlA01.Status.ActPosition/10;

END_IF

/*************************************************************/

IF gBasicControlA01.AxisState.MoveAbsoluteDone THEN
gBasicControlA01.Command.MoveAbsolute :=FALSE;

/*****************************************************************
***********          Hübe zählen                     ***************
*****************************************************************/

IF iCounter <= uHub THEN
```

```
iCounter                    := iCounter + 1;
uStateMachine    := 3000;

ELSE
/*****************************************************************
************          Test beenden                  ****************
*****************************************************************/

uStateMachine    := 3030;
iCounter                    := 0;
bStepEnable                 := FALSE;

END_IF

END_IF

3030: /******************          Test fertig              ********************/

/*****************************************************************
************     Zum Manuellen Betrieb              ****************
*****************************************************************/

IF NOT bAutoBetrieb THEN

uStateMachine    := 30;
bMove                       := FALSE;

END_IF

/*****************************************************************
************      Neuer (Drehmoment) Test           ****************
```

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

IF bNeueTest THEN

uStateMachine    := 30;
bNeueTest                := FALSE;
iCounter                 := 0;
bStepEnable              := FALSE;

END_IF


3040: /*****************  Zum Anfang Position nach Fehler  ******************/

gBasicControlA01.Parameter.Position                    := rStartPosition;
gBasicControlA01.Command.Halt                          := FALSE;
gBasicControlA01.Command.MoveAbsolute          := TRUE;

IF gBasicControlA01.AxisState.MoveAbsoluteDone THEN

gBasicControlA01.Command.MoveAbsolute          := FALSE;
gBasicControlA01.AxisState.MoveAbsoluteDone    := FALSE;
uStateMachine                                                  := 3050;

END_IF

3050: /*****************     Fehler Zustand              ********************/

/*****************************************************************
************         Zum Manuellen Betrieb          ****************
*****************************************************************/
```

```
IF bTestAbbrechen THEN

uStateMachine    := 20;
iCounter                 := 0;
bAutoBetrieb     := FALSE;
bTestAbbrechen   := FALSE;
bStepEnable              := FALSE;

END_IF

/*****************************************************************
***********        Ende Position ändern              ***************
*****************************************************************/

IF bErrorEnd THEN

bStepEnable      := FALSE;
rEndPosition     := rEndPosition + iWinkelKorr*10;
uStateMachine    := 3000;
bErrorEnd                := FALSE;

END_IF

5000: //Fehler

gBasicControlA01.Command.Power                          := FALSE;
gBasicControlA01.Command.Home                           := FALSE;
gBasicControlA01.Command.MoveAbsolute           := FALSE;
gBasicControlA01.Command.MoveAdditive           := FALSE;
gBasicControlA01.Command.MoveVelocity           := FALSE;
gBasicControlA01.Command.Halt                           := TRUE;
gBasicControlA01.Command.Stop                           := FALSE;
```

```
gBasicControlA01.Command.MoveJogPos                    := FALSE;
gBasicControlA01.Command.MoveJogNeg                    := FALSE;

IF bIndFinden THEN

uStateMachine                                                  := 0;
bIndBypass                                                             := TRUE;
bIndFinden                                                             := FALSE;
gBasicControlA01.Command.Halt                  := FALSE;

END_IF

END_CASE

/*****************************************************************************
******************** Frühere Werte speichern für Filter*********************
*****************************************************************************/

//Kraft

kxn10           := kxn9;
kxn9            := kxn8;
kxn8            := kxn7;
kxn7            := kxn6;
kxn6            := kxn5;
kxn5            := kxn4;
kxn4            := kxn3;
kxn3            := kxn2;
kxn2            := kxn1;
kxn1            := rKraftIst150kN;

kyn1            := rKraftFiltriert150kN;
```

```
k2xn10              := k2xn9;
k2xn9               := k2xn8;
k2xn8               := k2xn7;
k2xn7               := k2xn6;
k2xn6               := k2xn5;
k2xn5               := k2xn4;
k2xn4               := k2xn3;
k2xn3               := k2xn2;
k2xn2               := k2xn1;
k2xn1               := rKraftIst40kN;

k2yn1               := rKraftFiltriert40kN;

//Drehoment

mxn10               := mxn9;
mxn9                := mxn8;
mxn8                := mxn7;
mxn7                := mxn6;
mxn6                := mxn5;
mxn5                := mxn4;
mxn4                := mxn3;
mxn3                := mxn2;
mxn2                := mxn1;
mxn1                := rMn1;

myn1                := rMoment;

//Drehmomentableitung

rDerMn10            := rDerMn9;
```

```
rDerMn9             := rDerMn8;
rDerMn8             := rDerMn7;
rDerMn7             := rDerMn6;
rDerMn6             := rDerMn5;
rDerMn5             := rDerMn4;
rDerMn4             := rDerMn3;
rDerMn3             := rDerMn2;
rDerMn2             := rDerMn1;
rDerMn1             := rDerMp;

rDerMyn1            := rDerM;

/*********************************************************************
******************** Regelungsprozess ********************************
*********************************************************************/

CASE uMod OF

1:
rLetzteVal          := rMaxKraft;

3:
rLetzteVal          := rMaxMoment;

END_CASE

END_PROGRAM

PROGRAM _INIT

/* get axis object */
Axis1Obj := ADR(gAxis01);
```

```
    AxisStep := STATE_WAIT; /* start step */

END_PROGRAM
```

E.2. basicCycle.st

```
/********************************************************************
* COPYRIGHT -- Bernecker + Rainer
********************************************************************
* PROGRAM: Basic
* File: basicCyclic.st
* Author: Bernecker + Rainer
* Created: December 01, 2009
********************************************************************
* Implementation of Program Basic
********************************************************************/
PROGRAM _CYCLIC

BasicControl.Command                        := gBasicControlA01.Command;
BasicControl.Parameter                      := gBasicControlA01.Parameter;
BasicControl.Parameter.JogVelocity      := 250;
BasicControl.Parameter.Velocity         := 1620;



/********************************************************************
Control Sequence
********************************************************************/
/* status information is read before the step sequencer to attain a shorter reaction time */
/********************** MC_READSTATUS ************************/
MC_ReadStatus_0.Enable := NOT(MC_ReadStatus_0.Error);
```

```
MC_ReadStatus_0.Axis := Axis1Obj;
MC_ReadStatus_0();
BasicControl.AxisState.Disabled             := MC_ReadStatus_0.Disabled;
BasicControl.AxisState.StandStill           := MC_ReadStatus_0.StandStill;
BasicControl.AxisState.Stopping             := MC_ReadStatus_0.Stopping;
BasicControl.AxisState.Homing               := MC_ReadStatus_0.Homing;
BasicControl.AxisState.DiscreteMotion       := MC_ReadStatus_0.DiscreteMotion;
BasicControl.AxisState.ContinuousMotion     := MC_ReadStatus_0.ContinuousMotion;
BasicControl.AxisState.SynchronizedMotion   := MC_ReadStatus_0.SynchronizedMotion;
BasicControl.AxisState.ErrorStop            := MC_ReadStatus_0.Errorstop;

/*******************MC_BR_READDRIVESTATUS**********************/
MC_BR_ReadDriveStatus_0.Enable := NOT(MC_BR_ReadDriveStatus_0.Error);
MC_BR_ReadDriveStatus_0.Axis := Axis1Obj;
MC_BR_ReadDriveStatus_0.AdrDriveStatus := ADR(BasicControl.Status.DriveStatus);
MC_BR_ReadDriveStatus_0();

/******************* MC_READACTUALPOSITION ********************/
MC_ReadActualPosition_0.Enable := (NOT(MC_ReadActualPosition_0.Error));
MC_ReadActualPosition_0.Axis := Axis1Obj;
MC_ReadActualPosition_0();
IF(MC_ReadActualPosition_0.Valid = TRUE)THEN
BasicControl.Status.ActPosition := MC_ReadActualPosition_0.Position;
END_IF

/******************* MC_READACTUALVELOCITY ********************/
MC_ReadActualVelocity_0.Enable := (NOT(MC_ReadActualVelocity_0.Error));
MC_ReadActualVelocity_0.Axis := Axis1Obj;
MC_ReadActualVelocity_0();
IF(MC_ReadActualVelocity_0.Valid = TRUE)THEN
BasicControl.Status.ActVelocity := MC_ReadActualVelocity_0.Velocity;
END_IF
```

```
/********************** MC_READAXISERROR *********************/
MC_ReadAxisError_0.Enable := NOT(MC_ReadAxisError_0.Error);
MC_ReadAxisError_0.Axis := Axis1Obj;
MC_ReadAxisError_0.DataAddress := ADR(BasicControl.Status.ErrorText);
MC_ReadAxisError_0.DataLength := SIZEOF(BasicControl.Status.ErrorText);
MC_ReadAxisError_0.DataObjectName := "acp10etxde";
MC_ReadAxisError_0();

/*************** CHECK FOR GENERAL AXIS ERROR *****************/
IF ((MC_ReadAxisError_0.AxisErrorID <> 0) AND (MC_ReadAxisError_0.Valid = TRUE)) THEN
AxisStep := STATE_ERROR_AXIS;
/***************** CHECK IF POWER SHOULD BE OFF ******************/
ELSIF ((BasicControl.Command.Power = FALSE) AND (MC_ReadAxisError_0.Valid = TRUE)) THEN
IF ((MC_ReadStatus_0.Errorstop = TRUE) AND (MC_ReadStatus_0.Valid = TRUE)) THEN
AxisStep := STATE_ERROR_RESET;
ELSE
AxisStep := STATE_WAIT;
END_IF
END_IF

/* central monitoring OF stop command attains a shorter reaction TIME in CASE OF emergency stop */
/*****************CHECK FOR STOP COMMAND************************/
IF (BasicControl.Command.Stop = TRUE) THEN
IF ((AxisStep >= STATE_HOME) AND (AxisStep <= STATE_ERROR)) THEN
/* reset all FB execute inputs we use */
MC_Home_0.Execute := 0;
MC_Stop_0.Execute := 0;
MC_MoveAbsolute_0.Execute := 0;
MC_MoveAdditive_0.Execute := 0;
MC_MoveVelocity_0.Execute := 0;
MC_ReadAxisError_0.Acknowledge := 0;
```

```
MC_Reset_0.Execute := 0;
MC_Halt_0.Execute := 0;

/* reset user commands */
BasicControl.Command.Halt := 0;
BasicControl.Command.Home := 0;
BasicControl.Command.MoveJogPos := 0;
BasicControl.Command.MoveJogNeg := 0;
BasicControl.Command.MoveVelocity := 0;
BasicControl.Command.MoveAbsolute := 0;
BasicControl.Command.MoveAdditive := 0;
AxisStep := STATE_STOP;
END_IF
END_IF

CASE AxisStep OF

/******************** WAIT ************************/
STATE_WAIT:   /* STATE: Wait */
IF (BasicControl.Command.Power = TRUE) THEN
AxisStep := STATE_POWER_ON;
ELSE
MC_Power_0.Enable := FALSE;
END_IF

/* reset all FB execute inputs we use */
MC_Home_0.Execute := FALSE;
MC_Stop_0.Execute := FALSE;
MC_MoveAbsolute_0.Execute := FALSE;
MC_MoveAdditive_0.Execute := FALSE;
MC_MoveVelocity_0.Execute := FALSE;
MC_Halt_0.Execute := FALSE;
```

```
MC_ReadAxisError_0.Acknowledge := FALSE;
MC_Reset_0.Execute := FALSE;

/* reset user commands */
BasicControl.Command.Stop := FALSE;
BasicControl.Command.Halt := FALSE;
BasicControl.Command.Home := FALSE;
BasicControl.Command.MoveJogPos := FALSE;
BasicControl.Command.MoveJogNeg := FALSE;
BasicControl.Command.MoveVelocity := FALSE;
BasicControl.Command.MoveAbsolute := FALSE;
BasicControl.Command.MoveAdditive := FALSE;

BasicControl.Status.ErrorID := 0;

/******************** POWER ON **********************/
STATE_POWER_ON:   /* STATE: Power on */
MC_Power_0.Enable := TRUE;
IF (MC_Power_0.Status = TRUE) THEN
AxisStep := STATE_READY;
END_IF
/* if a power error occured go to error state */
IF (MC_Power_0.Error = TRUE) THEN
BasicControl.Status.ErrorID := MC_Power_0.ErrorID;
AxisStep := STATE_ERROR;
END_IF

/******************** READY **********************/
STATE_READY:   /* STATE: Waiting for commands */
IF (BasicControl.Command.Home = TRUE)THEN
BasicControl.Command.Home := FALSE;
AxisStep := STATE_HOME;
```

```
ELSIF (BasicControl.Command.Stop = TRUE) THEN
AxisStep := STATE_STOP;

ELSIF (BasicControl.Command.MoveJogPos = TRUE) THEN
AxisStep := STATE_JOG_POSITIVE;

ELSIF (BasicControl.Command.MoveJogNeg = TRUE) THEN
AxisStep := STATE_JOG_NEGATIVE;

ELSIF (BasicControl.Command.MoveAbsolute = TRUE) THEN
BasicControl.Command.MoveAbsolute := FALSE;
AxisStep := STATE_MOVE_ABSOLUTE;

ELSIF (BasicControl.Command.MoveAdditive = TRUE) THEN
BasicControl.Command.MoveAdditive := FALSE;
AxisStep := STATE_MOVE_ADDITIVE;

ELSIF (BasicControl.Command.MoveVelocity = TRUE) THEN
BasicControl.Command.MoveVelocity := FALSE;
AxisStep := STATE_MOVE_VELOCITY;

ELSIF (BasicControl.Command.Halt = TRUE) THEN
BasicControl.Command.Halt := FALSE;
AxisStep := STATE_HALT;
END_IF

/******************** HOME **********************/
STATE_HOME:   /* STATE: start homing process */
MC_Home_0.Position := BasicControl.Parameter.HomePosition;
MC_Home_0.HomingMode := BasicControl.Parameter.HomeMode;
MC_Home_0.Execute := TRUE;
```

```
IF (MC_Home_0.Done = TRUE) THEN
BasicControl.AxisState.HomeDone := TRUE;
MC_Home_0.Execute := FALSE;
AxisStep := STATE_READY;
END_IF
/* if a homing error occured go to error state */
IF (MC_Home_0.Error = TRUE) THEN
MC_Home_0.Execute := FALSE;
BasicControl.Status.ErrorID := MC_Home_0.ErrorID;
AxisStep := STATE_ERROR;
END_IF

/**********************HALT_MOVEMENT**************************/
STATE_HALT:    /* STATE: Halt movement */
MC_Halt_0.Deceleration := BasicControl.Parameter.Deceleration;
MC_Halt_0.Execute := TRUE;
IF (MC_Halt_0.Done = TRUE) THEN
MC_Halt_0.Execute := FALSE;
AxisStep := STATE_READY;
END_IF

/* check if error occured */
IF (MC_Halt_0.Error = TRUE) THEN
BasicControl.Status.ErrorID := MC_Halt_0.ErrorID;
MC_Halt_0.Execute := FALSE;
AxisStep := STATE_ERROR;
END_IF

/********************** STOP MOVEMENT ************************/
STATE_STOP:   /* STATE: Stop movement */
MC_Stop_0.Deceleration := BasicControl.Parameter.Deceleration;
MC_Stop_0.Execute := TRUE;
```

```
/* if axis is stopped go to ready state */
IF ((MC_Stop_0.Done = TRUE) AND (BasicControl.Command.Stop = FALSE)) THEN
MC_Stop_0.Execute := FALSE;
AxisStep := STATE_READY;
END_IF
/* check if error occured */
IF (MC_Stop_0.Error = TRUE) THEN
BasicControl.Status.ErrorID := MC_Stop_0.ErrorID;
MC_Stop_0.Execute := FALSE;
AxisStep := STATE_ERROR;
END_IF

/******************** START JOG MOVEMENT POSITVE **********************/
STATE_JOG_POSITIVE:  /* STATE: Start jog movement in positive direction */
MC_MoveVelocity_0.Velocity       := BasicControl.Parameter.JogVelocity;
MC_MoveVelocity_0.Acceleration   := BasicControl.Parameter.Acceleration;
MC_MoveVelocity_0.Deceleration   := BasicControl.Parameter.Deceleration;
MC_MoveVelocity_0.Direction      := mcPOSITIVE_DIR;
MC_MoveVelocity_0.Execute := TRUE;
IF (BasicControl.Command.MoveJogPos = FALSE) THEN
MC_MoveVelocity_0.Execute := FALSE;
AxisStep := STATE_HALT;
END_IF
/* check if error occured */
IF (MC_MoveVelocity_0.Error = TRUE) THEN
BasicControl.Status.ErrorID := MC_MoveVelocity_0.ErrorID;
MC_MoveVelocity_0.Execute := FALSE;
AxisStep := STATE_ERROR;
END_IF

/******************** START JOG MOVEMENT NEGATIVE **********************/
STATE_JOG_NEGATIVE:  /* STATE: Start jog movement in negative direction */
```

```
MC_MoveVelocity_0.Velocity       := BasicControl.Parameter.JogVelocity;
MC_MoveVelocity_0.Acceleration   := BasicControl.Parameter.Acceleration;
MC_MoveVelocity_0.Deceleration   := BasicControl.Parameter.Deceleration;
MC_MoveVelocity_0.Direction          := mcNEGATIVE_DIR;
MC_MoveVelocity_0.Execute := TRUE;
IF (BasicControl.Command.MoveJogNeg = FALSE) THEN
MC_MoveVelocity_0.Execute := FALSE;
AxisStep := STATE_HALT;
END_IF
/* check if error occured */
IF (MC_MoveVelocity_0.Error = TRUE) THEN
BasicControl.Status.ErrorID := MC_MoveVelocity_0.ErrorID;
MC_MoveVelocity_0.Execute := FALSE;
AxisStep := STATE_ERROR;
END_IF

/******************** START ABSOLUTE MOVEMENT ********************/
STATE_MOVE_ABSOLUTE:  /* STATE: Start absolute movement */
MC_MoveAbsolute_0.Position       := BasicControl.Parameter.Position;
MC_MoveAbsolute_0.Velocity       := BasicControl.Parameter.Velocity;
MC_MoveAbsolute_0.Acceleration   := BasicControl.Parameter.Acceleration;
MC_MoveAbsolute_0.Deceleration   := BasicControl.Parameter.Deceleration;
MC_MoveAbsolute_0.Direction      := BasicControl.Parameter.Direction;
MC_MoveAbsolute_0.Execute := TRUE;
/* check if commanded position is reached */
IF (BasicControl.Command.Halt) THEN
BasicControl.Command.Halt := FALSE;
MC_MoveAbsolute_0.Execute := FALSE;
AxisStep := STATE_HALT;
END_IF

BasicControl.AxisState.MoveAbsoluteDone:=MC_MoveAbsolute_0.Done;
```

```
IF NOT BasicControl.Command.MoveAbsolute THEN
BasicControl.AxisState.MoveAbsoluteDone:=FALSE;

MC_MoveAbsolute_0.Execute := FALSE;
AxisStep := STATE_READY;
END_IF
/* check if error occured */
IF (MC_MoveAbsolute_0.Error = TRUE) THEN
BasicControl.Status.ErrorID := MC_MoveAbsolute_0.ErrorID;
MC_MoveAbsolute_0.Execute := FALSE;
AxisStep := STATE_ERROR;
END_IF

/******************** START ADDITIVE MOVEMENT ********************/
STATE_MOVE_ADDITIVE:  /* STATE: Start additive movement */
MC_MoveAdditive_0.Distance       := BasicControl.Parameter.Distance;
MC_MoveAdditive_0.Velocity       := BasicControl.Parameter.Velocity;
MC_MoveAdditive_0.Acceleration   := BasicControl.Parameter.Acceleration;
MC_MoveAdditive_0.Deceleration   := BasicControl.Parameter.Deceleration;
MC_MoveAdditive_0.Execute := TRUE;
/* check if commanded distance is reached */
IF (BasicControl.Command.Halt) THEN
BasicControl.Command.Halt := FALSE;
MC_MoveAdditive_0.Execute := FALSE;
AxisStep := STATE_HALT;
ELSIF (MC_MoveAdditive_0.Done = TRUE) THEN
MC_MoveAdditive_0.Execute := FALSE;
AxisStep := STATE_READY;
END_IF
/* check if error occured */
IF (MC_MoveAdditive_0.Error = TRUE) THEN
```

```
BasicControl.Status.ErrorID := MC_MoveAdditive_0.ErrorID;
MC_MoveAdditive_0.Execute := FALSE;
AxisStep := STATE_ERROR;
END_IF

/******************** START VELOCITY MOVEMENT ********************/
STATE_MOVE_VELOCITY:  /* STATE: Start velocity movement */
MC_MoveVelocity_0.Velocity       := BasicControl.Parameter.Velocity;
MC_MoveVelocity_0.Acceleration   := BasicControl.Parameter.Acceleration;
MC_MoveVelocity_0.Deceleration   := BasicControl.Parameter.Deceleration;
MC_MoveVelocity_0.Direction      := BasicControl.Parameter.Direction;
MC_MoveVelocity_0.Execute := TRUE;
/* check if commanded velocity is reached */
IF (BasicControl.Command.Halt) THEN
BasicControl.Command.Halt := FALSE;
MC_MoveVelocity_0.Execute := FALSE;
AxisStep := STATE_HALT;
ELSIF (MC_MoveVelocity_0.InVelocity = TRUE) THEN
MC_MoveVelocity_0.Execute := FALSE;
AxisStep := STATE_READY;
END_IF
/* check if error occured */
IF (MC_MoveVelocity_0.Error = TRUE) THEN
BasicControl.Status.ErrorID := MC_MoveVelocity_0.ErrorID;
MC_MoveVelocity_0.Execute := FALSE;
AxisStep := STATE_ERROR;
END_IF

/******************** FB-ERROR OCCURED ************************/
STATE_ERROR:  /* STATE: Error */
/* check if FB indicates an axis error */
IF (MC_ReadAxisError_0.AxisErrorCount<>0) THEN
```

```
AxisStep := STATE_ERROR_AXIS;
ELSE
IF (BasicControl.Command.ErrorAcknowledge = TRUE) THEN
BasicControl.Command.ErrorAcknowledge    := FALSE;
BasicControl.Status.ErrorID                    := 0;
/* reset axis if it is in axis state ErrorStop */
IF ((MC_ReadStatus_0.Errorstop = TRUE) AND (MC_ReadStatus_0.Valid = TRUE)) THEN
AxisStep := STATE_ERROR_RESET;
ELSE
AxisStep := STATE_WAIT;
END_IF
END_IF
END_IF

/******************** AXIS-ERROR OCCURED ************************/
STATE_ERROR_AXIS:  /* STATE: Axis Error */
IF (MC_ReadAxisError_0.Valid = TRUE) THEN
IF (MC_ReadAxisError_0.AxisErrorID <> 0) THEN
BasicControl.Status.ErrorID := MC_ReadAxisError_0.AxisErrorID;
END_IF
MC_ReadAxisError_0.Acknowledge := FALSE;
IF (BasicControl.Command.ErrorAcknowledge = TRUE) THEN
BasicControl.Command.ErrorAcknowledge := FALSE;
/* acknowledge axis error */
IF (MC_ReadAxisError_0.AxisErrorID <> 0) THEN
MC_ReadAxisError_0.Acknowledge := TRUE;
END_IF
END_IF
IF (MC_ReadAxisError_0.AxisErrorCount = 0) THEN
/* reset axis if it is in axis state ErrorStop */
BasicControl.Status.ErrorID                    := 0;
IF ((MC_ReadStatus_0.Errorstop = TRUE) AND (MC_ReadStatus_0.Valid = TRUE)) THEN
```

```
AxisStep := STATE_ERROR_RESET;
ELSE
AxisStep := STATE_WAIT;
END_IF
END_IF
END_IF

/******************** RESET DONE ************************/
STATE_ERROR_RESET:  /* STATE: Wait for reset done */
MC_Reset_0.Execute := TRUE;
/* reset MC_Power.Enable if this FB is in Error*/
IF (MC_Power_0.Error = TRUE) THEN
MC_Power_0.Enable := FALSE;
END_IF
IF(MC_Reset_0.Done = TRUE)THEN
MC_Reset_0.Execute                                          := FALSE;
AxisStep := STATE_WAIT;
ELSIF(MC_Reset_0.Error = TRUE) THEN
MC_Reset_0.Execute := FALSE;
AxisStep := STATE_ERROR;
END_IF
/******************** SEQUENCE END ************************/
END_CASE

/*************************************************************
Function Block Calls
*************************************************************/

/************************** MC_POWER ***************************/
MC_Power_0.Axis := Axis1Obj;  /* pointer to axis */
MC_Power_0();
```

```
/************************ MC_HOME ****************************/
MC_Home_0.Axis := Axis1Obj;
MC_Home_0();

/********************* MC_MOVEABSOLUTE ************************/
MC_MoveAbsolute_0.Axis := Axis1Obj;
MC_MoveAbsolute_0();

/********************* MC_MOVEADDITIVE ************************/
MC_MoveAdditive_0.Axis := Axis1Obj;
MC_MoveAdditive_0();

/********************* MC_MOVEVELOCITY ************************/
MC_MoveVelocity_0.Axis := Axis1Obj;
MC_MoveVelocity_0();

/************************ MC_STOP ****************************/
MC_Stop_0.Axis := Axis1Obj;
MC_Stop_0();

/************************MC_HALT*****************************/
MC_Halt_0.Axis := Axis1Obj;
MC_Halt_0();

/************************ MC_RESET ***************************/
MC_Reset_0.Axis := Axis1Obj;
MC_Reset_0();

gBasicControlA01.AxisState      := BasicControl.AxisState;
gBasicControlA01.Status         := BasicControl.Status;

END_PROGRAM
```

E.3. DateiSpeichern.st

```
PROGRAM _INIT
/* Insert code here */
sZU:="$n";
END_PROGRAM

PROGRAM _CYCLIC

/*********       Offset rücksetzen        *********/
IF iCounter = 0 THEN
uOffsetn1        := 0;
END_IF

IF iCounter <= 1 THEN
uOffset          := 0;
END_IF

IF bStepEnable AND (iCounter <> uCounter) AND (rKraftHolder = 0) AND (rMHolder = 0) THEN

IF uCounter = 0 THEN
/*********** File erstellen in dem ersten Hub   *********/

uOffset                          := 0;
sDateTime                        := "";
sDateiName                       := "";

/***********     Datum in Filename hinzufügen    *********/

tDateTime.enable          := TRUE;
tDateTime();
```

```
DT_TO_DTStructure(tDateTime.DT1,ADR(tDate));

itoa(tDate.year,ADR(tHolder));
strcat(ADR(sDateTime),ADR(tHolder));

itoa(tDate.month,ADR(tHolder));

IF tDate.month < 10 THEN

strcat(ADR(sDateTime),ADR("0"));

END_IF

strcat(ADR(sDateTime),ADR(tHolder));

itoa(tDate.day,ADR(tHolder));

IF tDate.day < 10 THEN

strcat(ADR(sDateTime),ADR("0"));

END_IF

strcat(ADR(sDateTime),ADR(tHolder));

strcat(ADR(sDateiName),ADR(sDateTime));
strcat(ADR(sDateiName),ADR("_"));

/**********      Andere Datei in Filename hinzufügen      ********/

strcat(ADR(sDateiName),ADR(sKonstrukteur));
strcat(ADR(sDateiName),ADR("_"));
```

```
strcat(ADR(sDateiName),ADR(sVersuch));

/**********      Spanner Name hinzufügen (wenn verfügbar ********/

IF NOT bParMode THEN

strcat(ADR(sDateiName),ADR("_"));
strcat(ADR(sDateiName),ADR(sSpanner));

END_IF

/******** Filename ändern wenn er schon existiert ***********
********* FUNKTIONIERT NUR EINMAL!!!!!!!!!!!!!!!!  **********/

IF bNameEx THEN

strcat(ADR(sDateiName),ADR("(1)"));
bNameEx             := FALSE;

END_IF

strcat(ADR(sDateiName),ADR(".txt"));

bFileErstellen              := TRUE;

END_IF

/******** Datei in Text speichern **********/

ftoa(rKraftErreicht,ADR(sHolder));
strcat(ADR(sHolder),ADR(", "));
```

```
ftoa(rMomentErreicht,ADR(sEHolder));
strcat(ADR(sHolder),ADR(sEHolder));
strcat(ADR(sHolder),ADR(", "));

IF bKV THEN

ftoa(rKMomentErreicht,ADR(sEHolder));
strcat(ADR(sHolder),ADR(sEHolder));
strcat(ADR(sHolder),ADR(", "));

END_IF

ftoa(rWinkelErreicht,ADR(sEHolder));
strcat(ADR(sHolder),ADR(sEHolder));
strcat(ADR(sHolder),ADR(", "));

IF bKV THEN

ftoa(rPosKup1,ADR(sEHolder));
strcat(ADR(sHolder),ADR(sEHolder));
strcat(ADR(sHolder),ADR(", "));

ftoa(rPosKup2,ADR(sEHolder));
strcat(ADR(sHolder),ADR(sEHolder));
strcat(ADR(sHolder),ADR(", "));

END_IF

ftoa(rStartPos,ADR(sEHolder));
strcat(ADR(sHolder),ADR(sEHolder));
```

```
/************ Regelungsparameter schreiben *********/

CASE uMod OF

1:
strcat(ADR(sHolder),ADR(", Kmax_"));
ftoa(rMaxKraft,ADR(sEHolder));
strcat(ADR(sHolder),ADR(sEHolder));
2:
strcat(ADR(sHolder),ADR(", Winkel_"));
itoa(iWinkelPosReg,ADR(sEHolder));
strcat(ADR(sHolder),ADR(sEHolder));
3:
strcat(ADR(sHolder),ADR(", Mmax_"));
ftoa(rMaxMoment,ADR(sEHolder));
strcat(ADR(sHolder),ADR(sEHolder));
END_CASE

/********* Zeilenumbruch **************/

strcat(ADR(sHolder),ADR(sZU));

/********* Offset ändern **************/

IF iCounter >= 1 THEN

uOffset := uOffsetn1;
bDatei          := TRUE;

IF iCounter >= 2 THEN

uOffsetn1       := uOffset + strlen(ADR(sHolder));
```

```
END_IF

END_IF

uCounter          := iCounter;

END_IF


END_PROGRAM


PROGRAM _CYCLIC

bLeistungEin      := TRUE;
bNotAusOut1               := TRUE;
bNotAusOut2               := TRUE;
bHubSaul                  := NOT bStepEnable;

IF NOT (bNotAusIn1) OR (NOT bNotAusIn2) OR (uStateMachine = 5000) OR (gBasicControlA01.Status.
    ErrorID <> 0) OR ((NOT bHaube1 OR NOT bHaube2) AND NOT bHaubeAuf) OR bErrorState OR bFileError
   THEN
bAmpelRot         := TRUE;
bAmpelGruen       := FALSE;
bNotAus           := FALSE;
ELSE
bAmpelRot         := FALSE;
bAmpelGruen       := TRUE;
bNotAus           := TRUE;
END_IF
```

```
IF ((uStateMachine < 1000) OR (uStateMachine = 1030) OR (uStateMachine = 2030) OR (uStateMachine =
    3030) OR (uStateMachine = 1050) OR (uStateMachine = 2050) OR (uStateMachine = 3050)) THEN
bHaubeAuf        := TRUE;
ELSE
bHaubeAuf        := FALSE;
END_IF

IF (uStateMachine < 1000) OR (uStateMachine = 1030) OR (uStateMachine = 2030) OR (uStateMachine =
   3030) THEN
bVisuPow         := TRUE;
ELSE
bVisuPow         := FALSE;
END_IF

END_PROGRAM
```

E.4. FileHandling.st

```
/********************************************************************
* COPYRIGHT -- Bernecker + Rainer
********************************************************************
* Program: FileHandling
* File: FileHandling.st
* Author: Bernecker + Rainer
* Created: April 16, 2009
********************************************************************
* Implementation of program Handling
********************************************************************/

/*****************************************/
/* init program                          */
/*****************************************/
PROGRAM _INIT
```

```
/* here you can select your filedevice for the use in the cyclic part of this program*/
Select := 0;  /* local device e.g. compact flash*/
/*Select := 1;*/  /* USB device e.g. on IF6.ST1 of a X20CP1484 */

IF Select = 0 THEN
/* device name for different function blocks needed (choose yourself) */
strcpy( ADR(Handling.Data.Device), ADR("STAND") );
/* perameter string, layout see in the online help by function block "devlink" */
strcpy( ADR(Handling.Data.Parameter), ADR(""/DEVICE=F:") );
ELSIF Select = 1 THEN
/* device name for different function blocks needed (choose yourself) */
strcpy( ADR(Handling.Data.Device), ADR("USB_DEVICE") );
/* perameter string, layout see in the online help by function block "devlink" */
strcpy( ADR(Handling.Data.Parameter), ADR(""/DEVICE=IF6.ST1") );
END_IF

/* predefine filenames */
strcpy( ADR(Handling.Data.FileName), ADR("TestFile.txt") );
strcpy( ADR(Handling.Data.NewFileName), ADR("CopyTestFile.txt") );
/* predefine writedata */

END_PROGRAM

/*****************************************/
/* cyclic program                        */
/*****************************************/
PROGRAM _CYCLIC

IF NOT bFileErstellen THEN

Handling.Command.bCreateFile      := FALSE;
```

```
END_IF

IF bFileErstellen THEN

strcpy(ADR(Handling.Data.FileName),ADR(sDateiName));
Handling.Command.bCreateFile      := TRUE;
bFileErstellen                                    := FALSE;

END_IF

IF bDatei THEN

strcpy( ADR(Handling.Data.WriteData), ADR(sHolder) );
Handling.Functionblock.FileWrite_0.offset := uOffset;
Handling.Command.bWriteFile       := TRUE;
bDatei := FALSE;

END_IF



CASE Handling.Data.Step OF

0:  /* link (create) a file device */
Handling.Functionblock.DevLink_0.enable := 1;
Handling.Functionblock.DevLink_0.pDevice := ADR(Handling.Data.Device);  /* name of the device,
   which is needed for other functionblocks */
Handling.Functionblock.DevLink_0.pParam := ADR(Handling.Data.Parameter);  /* devicepath */
Handling.Functionblock.DevLink_0;  /* call the function*/

IF Handling.Functionblock.DevLink_0.status = 0 THEN  /* DevLink successful */
```

```
Handling.Data.Step := 1;  /* next Step*/
ELSIF Handling.Functionblock.DevLink_0.status = ERR_FUB_BUSY THEN  /* DevLink not finished -> redo
    */
/* Busy */
ELSIF Handling.Functionblock.DevLink_0.status = fiERR_SYSTEM THEN  /* DevLink error = fiERR_SYSTEM
    -> detailed info with function "FileIoGetSysError" */
/* get detail errorinformation */
Error := FileIoGetSysError();
/* Goto Error Step */
Handling.Data.Step := 255;
ELSE  /* Goto Error Step */
Handling.Data.Step := 255;
END_IF

1:  /* Wait and comannd step */
IF Handling.Command.bCreateFile = 1 THEN
/* Create a new file with the name from the variable "Handling.Data.FileName" */
Handling.Data.Step := 10;  /* next Step*/

ELSIF Handling.Command.bWriteFile = 1 THEN
/* writes the data from variable "Handling.Data.WriteData"
in the file with the name from the variable "Handling.Data.FileName" */
Handling.Data.Step := 20;  /* next Step*/

ELSIF Handling.Command.bReadFile = 1 THEN
/* read data to variable "Handling.Data.ReadData"
from the file with the name of the variable "Handling.Data.FileName" */
Handling.Data.Step := 30;  /* next Step*/

ELSIF Handling.Command.bReadExFile = 1 THEN
/* read data to variable "Handling.Data.ReadData"
from the file with the name of the variable "Handling.Data.FileName" */
```

```
Handling.Data.Step := 40;  /* next Step*/

ELSIF Handling.Command.bCopyFile = 1 THEN
/* creates a copy with the name from the variable "Handling.Data.NewFileName"
from the file with the name from the variable "Handling.Data.FileName" */
Handling.Data.Step := 50;  /* next Step*/

ELSIF Handling.Command.bRenameFile = 1 THEN
/* renames the the file with the name from the variable "Handling.Data.FileName"
to the name from the variable "Handling.Data.NewFileName" */
Handling.Data.Step := 60;  /* next Step*/

ELSIF Handling.Command.bDeleteFile = 1 THEN
/* deletes the file with the name from the variable "Handling.Data.FileName" */
Handling.Data.Step := 70;  /* next Step*/

END_IF

/*********************************************************************************************/
10:  /* create a new File with the selected name  */
Handling.Functionblock.FileCreate_0.enable := 1;
Handling.Functionblock.FileCreate_0.pDevice := ADR(Handling.Data.Device);  /* name of the linked
   device */
Handling.Functionblock.FileCreate_0.pFile := ADR(Handling.Data.FileName);  /* name of the file */
Handling.Functionblock.FileCreate_0;  /* call the function*/

IF Handling.Functionblock.FileCreate_0.status = 0 THEN  /* FileCreate successful */
Handling.Data.Step := 11;  /* next Step*/
ELSIF Handling.Functionblock.FileCreate_0.status = ERR_FUB_BUSY THEN  /* FileCreate not finished
   -> redo */
/* Busy */
ELSIF Handling.Functionblock.FileCreate_0.status = 20705 THEN
```

```
bNameEx := TRUE;
Handling.Data.Step       := 255;
ELSE  /* Goto Error Step */
Handling.Data.Step       := 255;
END_IF

11:      /* close file, because of limited number of available file handles on the system */
Handling.Functionblock.FileClose_0.enable := 1;
Handling.Functionblock.FileClose_0.ident := Handling.Functionblock.FileCreate_0.ident; /* ident
   for FileCreate-functionblock*/
Handling.Functionblock.FileClose_0;  /* call the function*/

IF Handling.Functionblock.FileClose_0.status = 0 THEN  /* FileClose successful */
Handling.Data.Step := 1;  /* next Step*/
Handling.Command.bCreateFile := 0;  /* clear command */
ELSIF Handling.Functionblock.FileClose_0.status = ERR_FUB_BUSY THEN  /* FileClose not finished ->
   redo */
/* Busy */
ELSE  /* Goto Error Step */
Handling.Data.Step := 255;
END_IF

/*****************************************************************************************/
20:     /* open file for write access */
Handling.Functionblock.FileOpen_0.enable := 1;
Handling.Functionblock.FileOpen_0.pDevice := ADR(Handling.Data.Device);  /* name of the linked
   device */
Handling.Functionblock.FileOpen_0.pFile :=  ADR(Handling.Data.FileName);  /* name of the file */
Handling.Functionblock.FileOpen_0.mode := fiWRITE_ONLY; /* write access */
Handling.Functionblock.FileOpen_0;  /* call the function*/

IF Handling.Functionblock.FileOpen_0.status = 0 THEN  /* FileOpen successful */
```

```
Handling.Data.Step := 21;  /* next Step*/
ELSIF Handling.Functionblock.FileOpen_0.status = ERR_FUB_BUSY THEN  /* FileOpen not finished ->
   redo */
/* Busy */
ELSE  /* Goto Error Step */
Handling.Data.Step := 255;
END_IF

21: /* write data into file */
Handling.Functionblock.FileWrite_0.enable := 1;
Handling.Functionblock.FileWrite_0.ident := Handling.Functionblock.FileOpen_0.ident;
Handling.Functionblock.FileWrite_0.pSrc := ADR(Handling.Data.WriteData);
Handling.Functionblock.FileWrite_0.len := strlen( ADR(Handling.Data.WriteData) );
Handling.Functionblock.FileWrite_0;  /* call the function*/

IF Handling.Functionblock.FileWrite_0.status = 0 THEN  /* FileWrite successful */
Handling.Data.Step := 22;  /* next Step*/
ELSIF Handling.Functionblock.FileWrite_0.status = ERR_FUB_BUSY THEN  /* FileWrite not finished ->
   redo */
/* Busy */
ELSE  /* Goto Error Step */
Handling.Data.Step := 255;
END_IF

22:      /* close file, because of limited number of available file handles on the system */
Handling.Functionblock.FileClose_0.enable := 1;
Handling.Functionblock.FileClose_0.ident := Handling.Functionblock.FileOpen_0.ident; /* ident for
   FileCreate-functionblock*/
Handling.Functionblock.FileClose_0;  /* call the function*/

IF Handling.Functionblock.FileClose_0.status = 0 THEN  /* FileClose successful */
Handling.Data.Step := 1;  /* next Step*/
```

```
Handling.Command.bWriteFile := 0;  /* clear command */
ELSIF Handling.Functionblock.FileClose_0.status = ERR_FUB_BUSY THEN  /* FileClose not finished ->
   redo */
/* Busy */
ELSE  /* Goto Error Step */
Handling.Data.Step := 255;
END_IF

/*****************************************************************************************/
30:     /* open file for read access */
Handling.Functionblock.FileOpen_0.enable := 1;
Handling.Functionblock.FileOpen_0.pDevice := ADR(Handling.Data.Device);  /* name of the linked
   device */
Handling.Functionblock.FileOpen_0.pFile :=  ADR(Handling.Data.FileName);  /* name of the file */
Handling.Functionblock.FileOpen_0.mode := fiREAD_ONLY;  /* read access */
Handling.Functionblock.FileOpen_0;  /* call the function*/

IF Handling.Functionblock.FileOpen_0.status = 0 THEN  /* FileOpen successful */
Handling.Data.Step := 31;  /* next Step*/
ELSIF Handling.Functionblock.FileOpen_0.status = ERR_FUB_BUSY THEN  /* FileOpen not finished ->
   redo */
/* Busy */
ELSE  /* Goto Error Step */
Handling.Data.Step := 255;
END_IF

31: /* read data from file */
Handling.Functionblock.FileRead_0.enable := 1;
Handling.Functionblock.FileRead_0.ident := Handling.Functionblock.FileOpen_0.ident;  /* ident of
   the previous "FileOpen" */
Handling.Functionblock.FileRead_0.offset := 0;  /* start at the beginning of the file */
Handling.Functionblock.FileRead_0.pDest := ADR(Handling.Data.ReadData);  /* adress of the
```

```
    destination of reaaded datas */
Handling.Functionblock.FileRead_0.len := SIZEOF( Handling.Data.ReadData );  /* lenght of data,
    which should be readed */
Handling.Functionblock.FileRead_0;  /* call the function*/

IF Handling.Functionblock.FileRead_0.status = 0 THEN  /* FileRead successful */
Handling.Data.Step := 32;  /* next Step*/
ELSIF Handling.Functionblock.FileRead_0.status = ERR_FUB_BUSY THEN  /* FileRead not finished ->
    redo */
/* Busy */
ELSE  /* Goto Error Step */
Handling.Data.Step := 255;
END_IF

32:      /* close file, because of limited number of available file handles on the system */
Handling.Functionblock.FileClose_0.enable := 1;
Handling.Functionblock.FileClose_0.ident := Handling.Functionblock.FileOpen_0.ident; /* ident for
    FileCreate-functionblock*/
Handling.Functionblock.FileClose_0;  /* call the function*/

IF Handling.Functionblock.FileClose_0.status = 0 THEN  /* FileClose successful */
Handling.Data.Step := 1;  /* next Step*/
Handling.Command.bReadFile := 0;  /* clear command */
ELSIF Handling.Functionblock.FileClose_0.status = ERR_FUB_BUSY THEN  /* FileClose not finished ->
    redo */
/* Busy */
ELSE  /* Goto Error Step */
Handling.Data.Step := 255;
END_IF

/*****************************************************************************************/
40:      /* open file for read access */
```

```
Handling.Functionblock.FileOpen_0.enable := 1;
Handling.Functionblock.FileOpen_0.pDevice := ADR(Handling.Data.Device);  /* name of the linked
   device */
Handling.Functionblock.FileOpen_0.pFile :=  ADR(Handling.Data.FileName);  /* name of the file */
Handling.Functionblock.FileOpen_0.mode := fiREAD_ONLY;  /* read access */
Handling.Functionblock.FileOpen_0;  /* call the function*/

IF Handling.Functionblock.FileOpen_0.status = 0 THEN  /* FileOpen successful */
Handling.Data.Step := 41;  /* next Step*/
ELSIF Handling.Functionblock.FileOpen_0.status = ERR_FUB_BUSY THEN  /* FileOpen not finished ->
   redo */
/* Busy */
ELSE  /* Goto Error Step */
Handling.Data.Step := 255;
END_IF

41: /* read data from file */
Handling.Functionblock.FileReadEx_0.enable := 1;
Handling.Functionblock.FileReadEx_0.ident := Handling.Functionblock.FileOpen_0.ident;  /* ident of
    the previous "FileOpen" */
Handling.Functionblock.FileReadEx_0.offset := 0;  /* start at the beginning of the file */
Handling.Functionblock.FileReadEx_0.pDest := ADR(Handling.Data.ReadData);  /* adress of the
   destination of reaaded datas */
Handling.Functionblock.FileReadEx_0.len := SIZEOF( Handling.Data.ReadData );  /* lenght of data,
   which should be readed */
Handling.Functionblock.FileReadEx_0;  /* call the function*/

IF Handling.Functionblock.FileReadEx_0.status = 0 THEN  /* FileReadEx successful */
/* Handling.Functionblock.FileReadEx_0.bytesread includes the lenght of the readed data */
Handling.Data.Step := 42;  /* next Step*/
ELSIF Handling.Functionblock.FileReadEx_0.status = ERR_FUB_BUSY THEN  /* FileReadEx not finished
   -> redo */
```

```
/* Busy */
ELSE  /* Goto Error Step */
Handling.Data.Step := 255;
END_IF

42:      /* close file, because of limited number of available file handles on the system */
Handling.Functionblock.FileClose_0.enable := 1;
Handling.Functionblock.FileClose_0.ident := Handling.Functionblock.FileOpen_0.ident; /* ident for
   FileCreate-functionblock*/
Handling.Functionblock.FileClose_0;  /* call the function*/

IF Handling.Functionblock.FileClose_0.status = 0 THEN  /* FileClose successful */
Handling.Data.Step := 1;  /* next Step*/
Handling.Command.bReadExFile := 0;  /* clear command */
ELSIF Handling.Functionblock.FileClose_0.status = ERR_FUB_BUSY THEN  /* FileClose not finished ->
   redo */
/* Busy */
ELSE  /* Goto Error Step */
Handling.Data.Step := 255;
END_IF

/*****************************************************************************************/
50: /* copy a file */

Handling.Functionblock.FileCopy_0.enable := 1;
Handling.Functionblock.FileCopy_0.option := fiOVERWRITE;  /* overwrite existing file */
Handling.Functionblock.FileCopy_0.pSrcDev := ADR(Handling.Data.Device);  /* name of a linked
   device */
Handling.Functionblock.FileCopy_0.pSrc := ADR(Handling.Data.FileName);  /* name of the source file
    */
Handling.Functionblock.FileCopy_0.pDestDev := ADR(Handling.Data.Device);  /* name of a linked
   device */
```

```
Handling.Functionblock.FileCopy_0.pDest := ADR(Handling.Data.NewFileName);  /* name of the new
    file */
Handling.Functionblock.FileCopy_0;  /* call the function */

IF Handling.Functionblock.FileCopy_0.status = 0 THEN  /* FileCopy successful */
Handling.Data.Step := 1;  /* next Step */
Handling.Command.bCopyFile := 0;  /* clear command */
ELSIF Handling.Functionblock.FileCopy_0.status = ERR_FUB_BUSY THEN  /* FileCopy not finished ->
    redo */
/* Busy */
ELSE  /* Goto Error Step */
Handling.Data.Step := 255;
END_IF

/*********************************************************************************************/
60: /* rename a file */

Handling.Functionblock.FileRename_0.enable := 1;
Handling.Functionblock.FileRename_0.pDevice := ADR(Handling.Data.Device);  /* name of the linked
    device */
Handling.Functionblock.FileRename_0.pName := ADR(Handling.Data.FileName);  /* actual file name */
Handling.Functionblock.FileRename_0.pNewName := ADR(Handling.Data.NewFileName);  /* new file name
    */
Handling.Functionblock.FileRename_0;  /* call the function */

IF Handling.Functionblock.FileRename_0.status = 0 THEN  /* FileRename successful */
Handling.Data.Step := 1;  /* next Step */
Handling.Command.bRenameFile := 0;  /* clear command */
ELSIF Handling.Functionblock.FileRename_0.status = ERR_FUB_BUSY THEN  /* FileRename not finished
    -> redo */
/* Busy */
ELSE  /* Goto Error Step */
```

```
Handling.Data.Step := 255;
END_IF

/*****************************************************************************************/
70: /* delete a file */

Handling.Functionblock.FileDelete_0.enable := 1;
Handling.Functionblock.FileDelete_0.pDevice := ADR(Handling.Data.Device);  /* name of a linked
   device */
Handling.Functionblock.FileDelete_0.pName := ADR(Handling.Data.FileName);  /* name of the source
   file */
Handling.Functionblock.FileDelete_0;  /* call the function*/

IF Handling.Functionblock.FileDelete_0.status = 0 THEN  /* FileRename successful */
Handling.Data.Step := 1;  /* next Step*/
Handling.Command.bDeleteFile := 0;  /* clear command */
ELSIF Handling.Functionblock.FileDelete_0.status = ERR_FUB_BUSY THEN  /* FileRename not finished
   -> redo */
/* Busy */
ELSE  /* Goto Error Step */
Handling.Data.Step := 255;
END_IF

/*****************************************************************************************/
255:

bFileError := TRUE; //Globaler Fehler Bezeichner

IF bNameEx THEN

Handling.Data.Step                  := 10;
Handling.Command.bWriteFile      := FALSE;
```

```
END_IF

IF  bFileErrorErk  THEN

Handling.Data.Step                  :=  1;
bFileError                                  :=  FALSE;

END_IF


END_CASE

END_PROGRAM
```

E.5. DateiLesenUndZeichnen.py

```
# -*- coding: utf-8 -*-
# coding: utf-8
"""
Created on Fri Aug 18 07:04:46 2017

@author: cad-9
"""


import numpy as np
import matplotlib.pyplot as plt
import xlsxwriter
#import xlwt
from matplotlib.backends.backend_pdf import PdfPages
import os
from ftplib import FTP
import re

ftp = FTP( host='192.168.27.101')
ftp.login(user='admin', passwd='1234')
ftp.cwd('F:/')
ls = []
data = []
def write(text):
ls.append(text)
return

ftp.retrlines('LIST',write)

for i in range(len(ls)):
print('%s: %s' %(i,ls[i]))

textfile = int(input('Wählen die Datei herunterzuladen '))

filename = ls[textfile].split()[8]
filename = filename.replace(".txt", "")

ftp.retrlines('RETR %s.txt' %filename ,data.append)


Kraft               =       np.empty([len(data)])
Drehmoment          =       np.empty([len(data)])
DMKupplung          =       np.empty([len(data)])
PosErreicht         =       np.empty([len(data)])
Kup1                =       np.empty([len(data)])
Kup2                =       np.empty([len(data)])
AnfangPos           =       np.empty([len(data)])
SteuerArt           =       []
SteuerArtReduziert  =       []
```

```
Hub                 =       np.linspace(1,len(data),len(data))

for i in range(len(data)):

holder      =       data[i].replace(",", "")
holder      =       holder.split()

if len(holder) == 8:

Kraft[i]         =    holder[0]
Drehmoment[i]    =    holder[1]
DMKupplung[i]    =    holder[2]
PosErreicht[i]   =    holder[3]
Kup1[i]          =    holder[4]
Kup2[i]          =    holder[5]
AnfangPos[i]     =    holder[6]
SteuerArt.append(holder[7])

else:

Drehmoment[i]    =    holder[0]
Kraft[i]         =    holder[1]
PosErreicht[i]   =    holder[2]
AnfangPos[i]     =    holder[3]
SteuerArt.append(holder[4])

PosErreicht=PosErreicht-AnfangPos

if len(holder) == 8:

Kup1 = Kup1 - AnfangPos
Kup2 = Kup2 - AnfangPos


Kz       = np.polyfit(np.linspace(1,len(Kraft),len(Kraft)), Kraft
    , 3)
Kfit     = []

for i in range(len(Kraft)):
Kneu     = Kz[0]*i**3 + Kz[1]*i**2 + Kz[2]*i + Kz[3]
Kfit.append(Kneu)

DMz       = np.polyfit(np.linspace(1,len(Drehmoment),len(
   Drehmoment)), Drehmoment , 4)
DMfit     = []

for i in range(len(Drehmoment)):
DMneu     = DMz[0]*i**4 + DMz[1]*i**3 + DMz[2]*i**2 + DMz[3]*i +
   DMz[4]
DMfit.append(DMneu)
```

```
DMKz       = np.polyfit(np.linspace(1,len(DMKupplung),len(
    DMKupplung)), DMKupplung, 4)
DMKfit     = []

for i in range(len(DMKupplung)):
DMKneu     = DMKz[0]*i**4 + DMKz[1]*i**3 + DMKz[2]*i**2 + DMKz
    [3]*i + DMKz[4]
DMKfit.append(DMKneu)

PEz        = np.polyfit(np.linspace(1,len(PosErreicht),len(
    PosErreicht)), PosErreicht, 3)
PEfit      = []

for i in range(len(PosErreicht)):
PEneu      = PEz[0]*i**3 + PEz[1]*i**2 + PEz[2]*i + PEz[3]
PEfit.append(PEneu)

K1z        = np.polyfit(np.linspace(1,len(Kup1),len(Kup1)), Kup1,
    2)
K1fit      = []

for i in range(len(Kup1)):
K1neu      = K1z[0]*i**2 + K1z[1]*i + K1z[2]
K1fit.append(K1neu)

K2z        = np.polyfit(np.linspace(1,len(Kup2),len(Kup2)), Kup2,
    2)
K2fit      = []

for i in range(len(Kup2)):
K2neu      = K2z[0]*i**2 + K2z[1]*i + K2z[2]
K2fit.append(K2neu)

with PdfPages("//FILESERVER/solid/Zeichnungen/Projekte/01 Fa.
    Allmatic/161220 Versuchsstand/10-Test-Ergebnisse/%s_Graphs.pdf
    " %filename) as pdf:

fig=plt.figure()
plt.plot(Hub,Kraft,'g',Hub,Kfit,'r',linewidth=0.8)
plt.xlabel('Hub')
plt.ylabel('Kraft (kN)')
plt.title('Kraft erreicht')
pdf.savefig()
plt.clf()
plt.plot(Hub,Drehmoment,'g',Hub,DMfit,'r',linewidth=0.8)
plt.xlabel('Hub')
plt.ylabel('Drehmoment (Nm)')
plt.title('Drehmoment erreicht')
pdf.savefig()

if len(holder) == 8:
```

```
plt.clf()
plt.plot(Hub,DMKupplung,'g',Hub,DMKfit,'r',linewidth=0.8)
plt.xlabel('Hub')
plt.ylabel('Drehmoment Nm')
plt.title('Drehmoment in Kupplung erreicht')
pdf.savefig()

plt.clf()
plt.plot(Hub,PosErreicht,'g',Hub,PEfit,'r',linewidth=0.8)
plt.ylabel('Winkel (°)')
plt.xlabel('Hub')
plt.title('Position Erreicht')
pdf.savefig()

if len(holder) == 8:

plt.clf()
plt.plot(Hub,Kup1,'g',Hub,K1fit,'r',linewidth=0.8)
plt.ylabel('Winkel (°)')
plt.xlabel('Hub')
plt.title('Kupplung Position 1')
pdf.savefig()
plt.clf()
plt.plot(Hub,Kup2,'g',Hub,K2fit,'r',linewidth=0.8)
plt.ylabel('Winkel (°)')
plt.xlabel('Hub')
plt.title('Kupplung Position 2 (Ausrasten)')
pdf.savefig()

#     plt.plot(Kraft/PosErreicht)
#     plt.xlabel('Hub')
#     plt.ylabel('Kraft (kN) / Winkel(°)')
#     plt.title('Kraft nach erreichte Position ')
#     pdf.savefig()

for i in range(len(SteuerArt)):
if SteuerArt[i] != SteuerArt[i-1]:
SteuerArtReduziert.append([SteuerArt[i-1],SteuerArt[i],i])
if len(SteuerArtReduziert) == 0:
SteuerArtReduziert.append(SteuerArt[0])

book     = xlsxwriter.Workbook('//FILESERVER/solid/Zeichnungen/
   Projekte/01 Fa. Allmatic/161220 Versuchsstand/10-Test-
   Ergebnisse/%s_Datei.xlsx'%filename)
sheet1   = book.add_worksheet('Datei')
sheet1.write(0,0,"Kraft")
sheet1.write(0,1,"Drehmoment")

if len(holder) == 8:
```

```
sheet1.write(0,2,"Drehmoment Kupplung")
sheet1.write(0,3,"Position erreicht")
sheet1.write(0,4,"Position Kupplung 1")
sheet1.write(0,5,"Position Kupplung 2")
sheet1.write(0,6,"Anfang Position")
sheet1.write(0,7,"Steuerungsart")

else:
sheet1.write(0,2,"Position erreicht")
sheet1.write(0,3,"Anfang Position")
sheet1.write(0,4,"Steuerungsart")

for i in range(len(data)):

n = i+1

sheet1.write(n,0,Kraft[i])
sheet1.write(n,1,Drehmoment[i])

if len(holder) == 8:

sheet1.write(n,2,DMKupplung[i])
sheet1.write(n,3,PosErreicht[i]+AnfangPos[i])
sheet1.write(n,4,Kup1[i]+AnfangPos[i])
sheet1.write(n,5,Kup2[i]+AnfangPos[i])
sheet1.write(n,6,AnfangPos[i])
sheet1.write(n,7,SteuerArt[i])

else:

sheet1.write(n,2,PosErreicht[i]+AnfangPos[i])
sheet1.write(n,3,AnfangPos[i])
sheet1.write(n,4,SteuerArt[i])

book.close()


os.startfile("\\\\FILESERVER\\solid\\Zeichnungen\\Projekte\\01
   Fa. Allmatic\\161220 Versuchsstand\\10-Test-Ergebnisse\\%
   s_Datei.xlsx" %filename)
os.startfile("\\\\FILESERVER\\solid\\Zeichnungen\\Projekte\\01
   Fa. Allmatic\\161220 Versuchsstand\\10-Test-Ergebnisse\\%
   s_Graphs.pdf" %filename)
```