Universitat de Vic
Escola Politècnica Superior

Trabajo Final de Carrera TFC

Sistema electrónico de calibración de fruta por diámetro

Erico Cruz Lemus

Ingeniería Técnica Industrial. Especialidad en Electrónica Industrial

Director: Pere Martí Puig

Vic, Octubre de 2012

UVIC Universitat de Vic
Escola Politècnica Superior

Resumen de Trabajo Final de Carrera TFC
Ingeniería Técnica Industrial Especialidad en Electrónica Industrial

Título: Sistema electrónico de calibración de fruta por diámetro
Palabras Clave: Visión artificial, calibradora, medición diámetro, clasificación de frutas, verduras y/o hortalizas, línea de producción alimentaria, cámaras infrarrojo, luz LED, básculas, cooperativas agricultoras.
Autor: Erico Cruz Lemus
Dirección: Pere Martí Puig
Fecha: Junio 2013

RESUMEN

En la actualidad, las cooperativas recolectan, seleccionan, tratan y separan la fruta según su calibre (peso, diámetro máximo, medio y/o mínimo) para que esta llegue al consumidor final según la categoría (calibre). Para poder competir en un mercado cada vez más exigente en calidad y precios, se requieren sistemas de clasificación automáticos que nos permitan obtener óptimos resultados con altos niveles de producción y productividad.

Para realizar estas tareas existen calibradoras industriales que pesan la fruta mediante células de carga y con el peso obtenido las clasifican asignando las piezas a su salida correspondiente (mesa de confección) a través de un sistema de electroimanes. Desafortunadamente el proceso de calibración de la fruta por peso no es en absoluto fiable ya que en este proceso no se considera el grosor de la piel, contenido de agua, de azúcar u otros factores altamente relevantes que influyen considerablemente en los resultados finales.

El objeto de este proyecto es el de evolucionar las existentes calibradoras de fruta instalando un sistema industrial de visión artificial (rápido y robusto) que trabaje en un rango de espectro Infrarrojo (mayor fiabilidad) proporcionando óptimos resultados finales en la clasificación de las frutas, verduras y hortalizas.

De este modo, el presente proyecto ofrece la oportunidad de mejorar el rendimiento de la línea de clasificación de frutas, aumentando la velocidad, disminuyendo pérdidas en tiempo y error humano y mejorando indiscutiblemente la calidad del producto final deseada por los consumidores.

UVIC Universitat de Vic
Escola Politècnica Superior

Final career work summary TFC
Industrial Technical Engineering Industrial Electronic Specialty

Title: Diameter fruit electronic calibration system
Key Works: Artificial vision, calibrator, diameter measurement, vegetables and fruits classification, food production line, infrared cameras, LED lights, load cells, scales, farmers cooperatives.
Author: Erico Cruz Lemus
Director: Pere Martí Puig
Date: June 2013

SUMMARY

At present, the cooperatives collect, select, treat and separate fruit according to its gauge (weight, maximum, medium or minimum diameter) to get this to the final consumer according to the category (caliber). To compete in a market more and more demanding in quality and prices, are required for automatic classification systems that will allow us to obtain excellent results with high levels of production and productivity.

To perform these tasks, there are sorting industrial machines weighting the fruit by load cells and sort the fruit by the weight obtained to the operator configured output (preparation table) through a set of solenoids. Unfortunately the calibration process by weight it is not at all reliable because this process do not consider the skin thickness, eater content, sugar other highly relevant factors that significantly influence the final results.

The purpose of this project is to evolve the existing fruits sorting systems by installing an industrial vision system machine (fast and robust) working in the infrared spectral range (more reliable) providing optimal final results in the fruits and vegetables classification process.

Thus, this project offers the opportunity to improve the performance of the fruit line classification, increasing speed, reducing losses in time and human error and improving the desired final product quality by consumers.

SUMMARY TFC

ÍNDICE

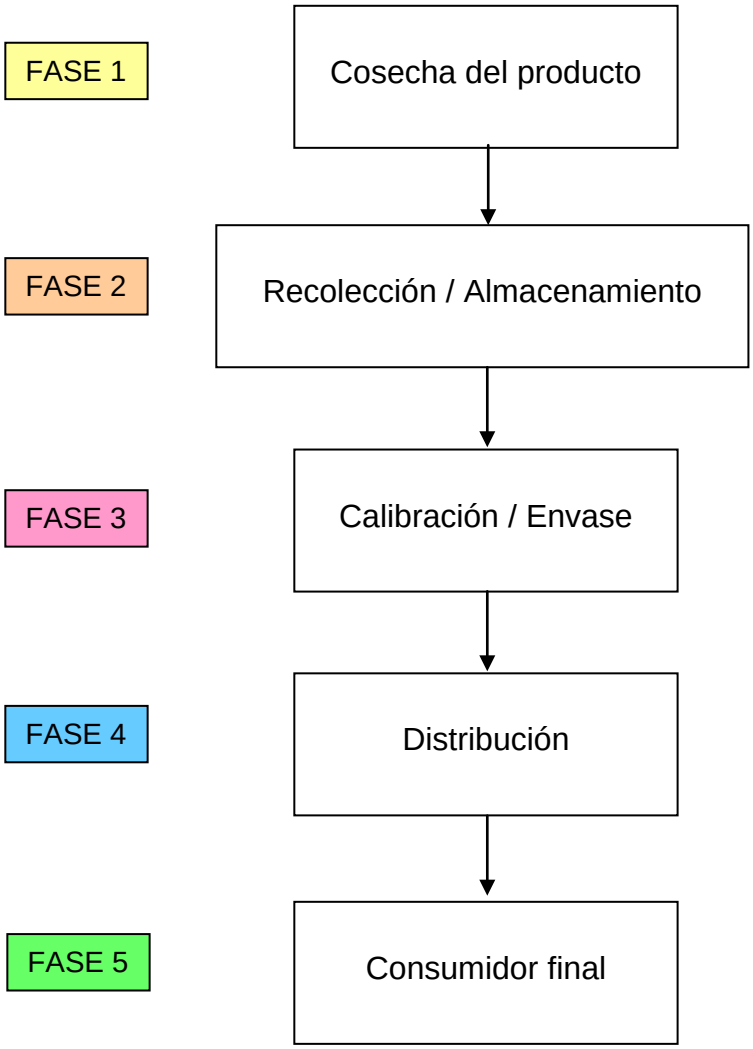
1. INTRODUCCIÓN

1.1. Antecedentes

Pocos somos conscientes de que el producto agroalimentario final que adquirimos en los establecimientos, las fruterías, antes de que llegue a manos del consumidor final, este ha tenido que ser procesado a través de diversas etapas cuyas actuales exigencias del mercado requieren de altos estándares de calidad y productividad (alta producción a un muy bajo costo).

1.1.1. Cadena del tratamiento y proceso de la fruta

Esta cadena de tratamiento y proceso de la fruta la podemos resumir de la siguiente manera:



Las cooperativas industriales que tratan y separan la fruta según su calibre (peso, color y diámetro), para poder competir en un mercado cada vez más exigente requieren de los más adecuados métodos para evitar quedarse atrás y por lo tanto enfrentarse a pérdidas nocivas para el negocio. Hoy en día la fruta se separa según su peso, color, defecto (daños visuales externos), diámetro e incluso por su contenido de azúcar, pero para poder procesar correctamente la fruta se requiere de un sistema altamente sofisticado cuya visión artificial será objeto de este proyecto.

1.2. Objeto del proyecto

Así pues, el objeto de este proyecto es el de diseñar un sistema de visión artificial que permita diferenciar la fruta según su diámetro proporcionando a la línea de producción las siguientes ventajas competitivas:

- Reducción de la mano de obra
- Altas velocidades en el proceso de calibración
- Fiabilidad (con un margen de error de +/- 1mm)

El objeto de este proyecto es el de diseñar un dispositivo que pueda trabajar de manera automática sin la necesidad de la presencia humana, de modo que la industria gane en producción y productividad además de mejorar el estándar de calidad del producto final gracias a una adecuada manipulación de la fruta y una reducción del error humano durante la adquisición de las medidas realizadas.

1.3. Especificaciones y abasto

El sector hortofrutícola, dependiendo del tipo de fruta que se procese, requiere poder seleccionar la fruta según su diámetro mínimo, máximo y/o medio.

Las funciones específicas que llevará a término la unidad de visión artificial diseñada en el presente proyecto serán las de capturar imágenes de las frutas y hortalizas que pasan por las citas de producción (hasta una velocidad máxima de 7 piezas por segundo), realizar los cálculos necesarios para determinar el diámetro mínimo, máximo y medio, y enviar los valores resultantes al computador principal que activará las salidas (por electroimán) que previamente haya configurado el operario de la máquina.

2. VISIÓN GLOBAL

2.1. Introducción

Tal y cómo hemos visto en el apartado “1.1.1. Cadena del tratamiento y proceso de la fruta”, existen diversas fases cuyas funciones son las de gestionar las frutas y hortalizas de manera eficiente para que dichos productos lleguen al alcance del consumidor final satisfactoriamente.

Como hemos mencionado en el apartado de especificaciones y abasto, nosotros nos centraremos en la FASE 3 (Calibración y almacenamiento), aunque lo que en realidad a nosotros nos concierne es la calibración de la fruta (en este caso, la calibración de la fruta por diámetro).

2.1.1. Antecedentes en la calibración de la fruta

Anteriormente, las frutas y hortalizas se calibraban manualmente utilizando diversos instrumentos como los de las siguientes ilustraciones:

Ilustración 1

Ilustración 2

Según el tipo de fruta y/o hortaliza que se desee procesar y el embase a utilizar, las piezas se seleccionan por diámetro máximo, mínimo, o diámetro medio (ver siguiente ilustración):

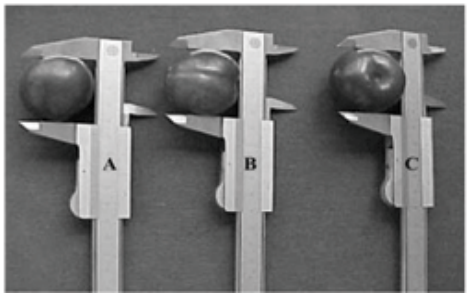


Ilustración 3

Donde la medición A mide el diámetro máximo, y las figuras B y C miden el diámetro mínimo.

2.1.2. Calibración por diámetro máximo

Un claro ejemplo de calibración por diámetro máximo lo podemos tener en alguna variedad de mandarinas, ya que estas son de forma achatada y necesitamos que en el embase, estas piezas queden derechas viéndose solo la parte superior (ver siguiente ilustración).

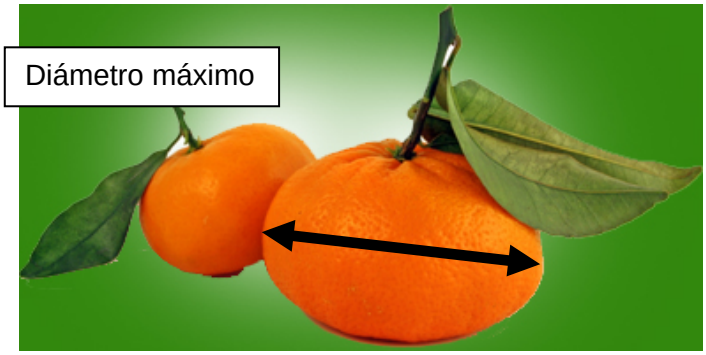


Ilustración 4

2.1.3. Calibración por diámetro mínimo

Sin embargo, para otros tipos de frutas y hortalizas con su embase utilizado para la salida al mercado, se requiere la medición del diámetro mínimo, como en el ejemplo de las dos siguientes ilustraciones:

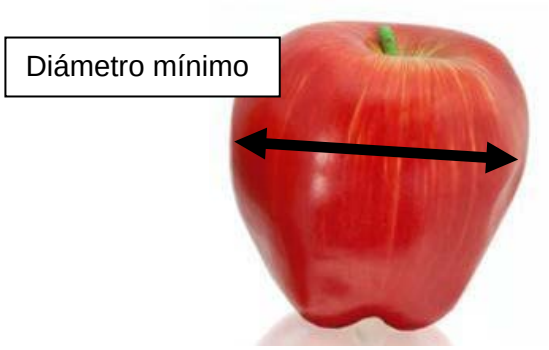


Ilustración 5

Ilustración 6

2.1.4. Calibración por diámetro medio

Por último, también se calibran las piezas, cuando estas son mucho mas redondas, utilizando el diámetro medio, por ejemplo con la naranjas de la siguiente ilustración.

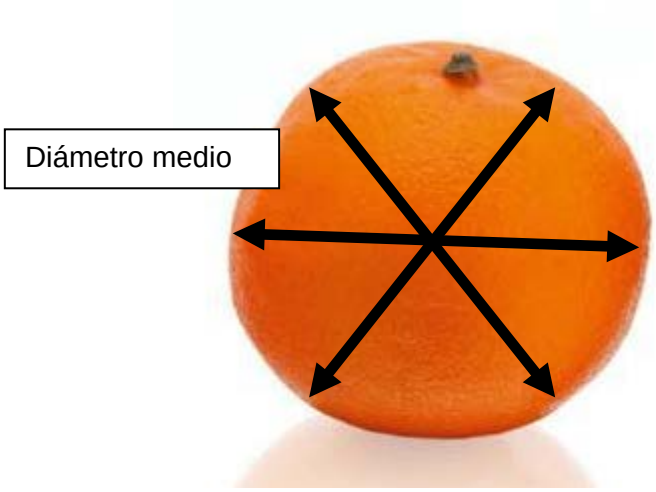


Ilustración 7

2.2. Proceso global calibración

Tal y como hemos mencionado anteriormente, el proceso de calibración se realizará a través de unas cámaras de visión artificial cuya instalación se realizará en línea con y de manera continua. En la siguiente ilustración se puede apreciar la línea global de calibración.

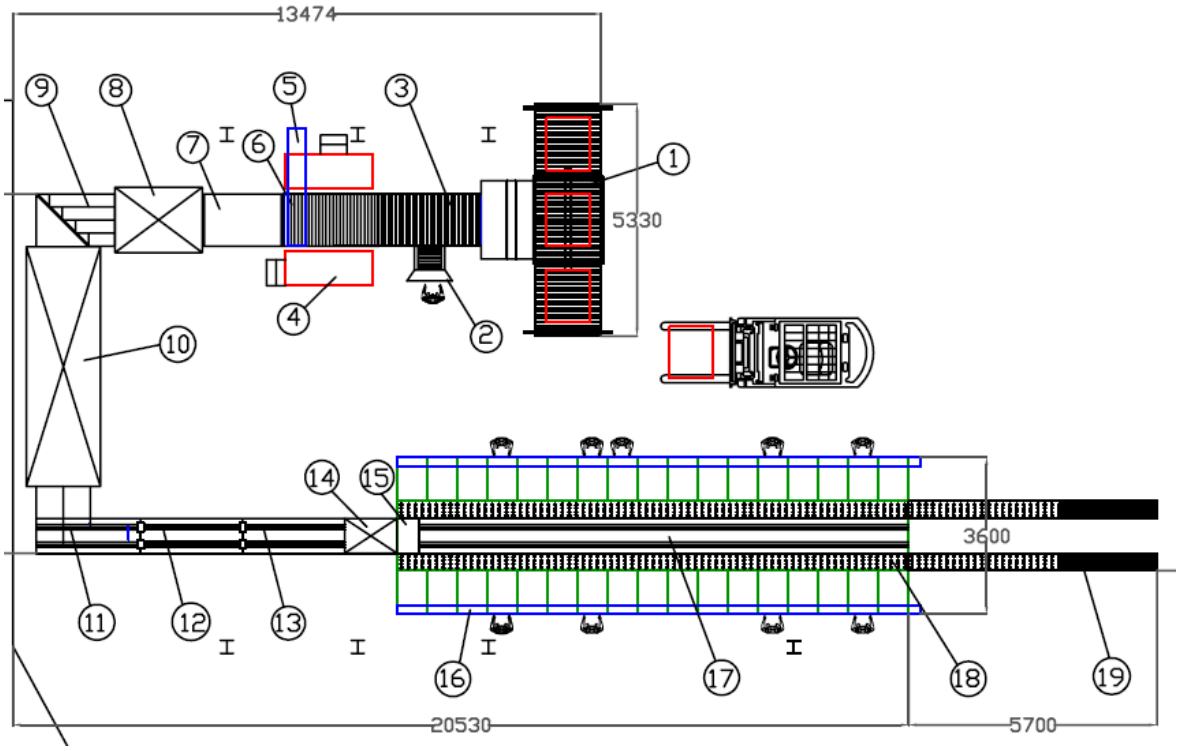


Ilustración 8

En la siguiente tabla se describe cada una de las partes de línea de calibración.

Pos.	Descripción	Unidades
1	VOLCADOR DE BINS	1
2	VOLCADOR MANUAL CAJAS DE LLENAS	1
3	TRAMO RODILLOS	1
4	PLATAFORMA ELEVADA	1
5	CINTA RETORNO DESTRIO	1
6	MESA DESTRIO	1
7	CEPILLADORA BYPASS	1
8	TÚNEL ENCERADO	1
9	TAPIZ MÚLTIPLE	1
10	TÚNEL SECADO	1
11	PRE-ALINEADOR	1
12	ALINEADOR	1
13	ROTACIÓN ENTRADAS CÁMARAS	1
14	**VISIÓN ARTIFICIAL**	**1**
15	CÉLULAS DE CARGA	1

16	MESAS DE CONFECCIÓN	2
17	CALIBRADORA 2 LINEAS, 16+1 SALIDAS	1
18	TRANSPORTADOR SALIDA DE CAJAS LLENAS	2
19	TRAMO DE RODILLOS	2

Tabla 1

(1) Volcador de bins:

En la anterior ilustración podemos comprender como la fruta llega desde las cámaras de refrigeración en el interior de unos recipientes industriales conocidos con el nombre de palots (ver siguiente ilustración)

Ilustración 9

Posteriormente estos palots son volcados semiautomáticamente de modo que las piezas entran en la línea de calibración.

(2) Volcador manual de cajas llenas:

En ocasiones, en la misma línea de producción se introducen las piezas manualmente en cajas convencionales.

(3) Tramo de rodillos y (6) Mesa de destrío:

En la zona “tramo de rodillos” diversos operarios extraen de la línea de calibración aquellas piezas que visualmente se consideran dañadas (destrío de la fruta defectuosa).

(4) Plataforma elevada:

Para efectuar el destrío de la fruta mencionado los operarios subirán a plataformas especialmente diseñadas para evitar accidentes y soportar el peso necesario.

(5) Cinta retorno destrío:

Mediante una cinta transportadora, la fruta destriada por los operarios retornará a un palot vacío cuyo contenido será enviado a plantas especializadas en el proceso y producción de zumos.

(7) Cepilladora Bypass:

Mediante unos cepillos de dureza especialmente diseñada para no dañar el género limpiamos la fruta.

(8) Túnel encerado:

Posteriormente mediante unos aspersores y unos cepillos se aplica una capa de cera especial y un pulido consiguiendo un brillo atractivo en la fruta para el consumidor final.

(9) Tapiz múltiple:

A través de un conjunto de tapices múltiples introducimos la fruta en el túnel de secado.

(10) Túnel de secado:

Con un sistema de ventiladores de alta presión y resistencias térmicas, secamos la fruta antes de que esta sea calibrada (ver siguiente ilustración).

Ilustración 10

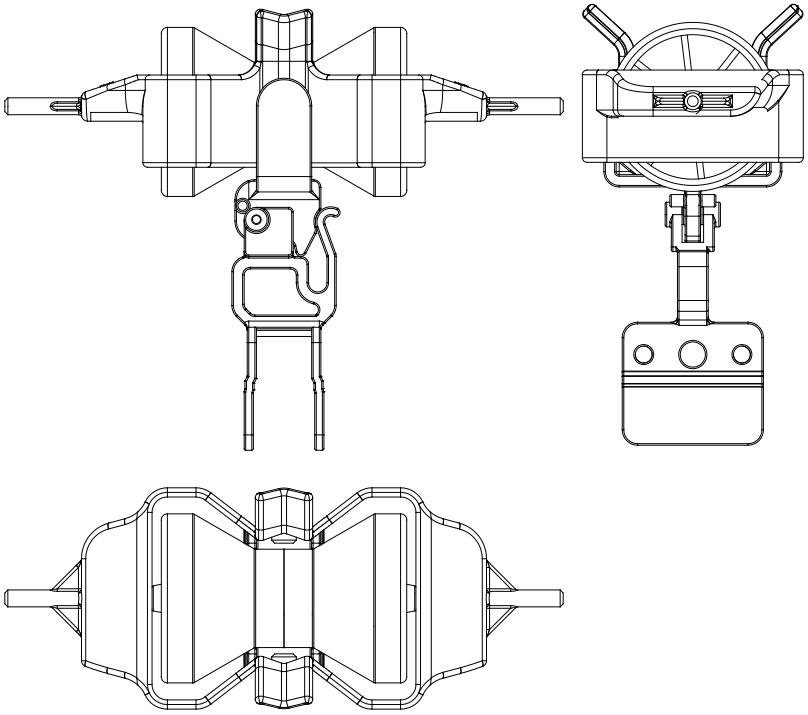
(11) Pre-alineador y (12) Alineador:

Un sistema de tapices funcionando a diversas velocidades se encargan de alinear la fruta para que esta llegue individualmente a la zona de visión artificial sin general falsas capturas de imágenes (ver siguiente ilustración).

Ilustración 11

(13) <u>Rotación entrada cámaras:</u>

Para conseguir captar imágenes en todo su perímetro, debe ir en un soporte que permita rotar las piezas mientras estas se desplazan horizontalmente. Para eso, hemos diseñado un sistema de manos o “cazoletas” que mediante unos rodillos y unas cintas transportadoras con velocidad lineal ajustable mediante un variador de frecuencia permite la rotación de la fruta y hortaliza requerida (ver siguiente ilustración).



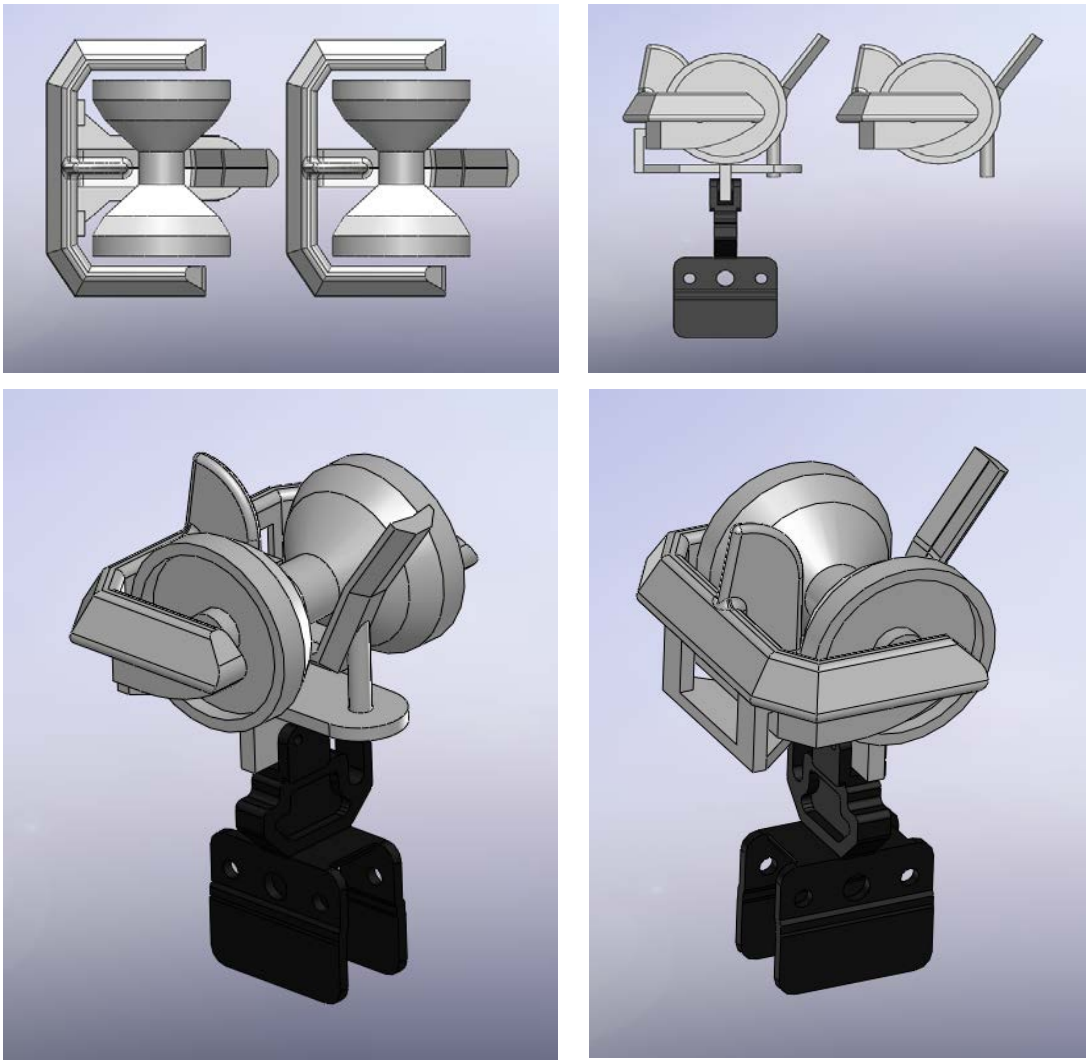


Ilustración 12

(14) <u>Visión artificial:</u>

En el módulo de visión artificial, alojaremos las cámaras y luces estroboscópicas que captarán las imágenes necesarias para realizar los cálculos de diámetro mencionados (ver siguiente ilustración).

Ilustración 13

(15) <u>Células de carga:</u>

Opcionalmente, en algunas líneas de calibración, podemos incluir un módulo de células de carga para adquirir los pesos de las piezas. De esto modo podemos calibrar la fruta con diámetro, con peso, o con peso y diámetro.

(16) <u>Mesas de confección:</u>

En las mesas de confección los operarios reciben la fruta calibrada para que puedan embasarla manualmente en las cajas especialmente diseñadas para ello (ver siguiente ilustración).

Ilustración 14

(17) <u>Calibradora:</u>

Después de obtener los resultados de tamaños de las piezas, mediante una cadena, sistema de “cazoletas” (antes mencionado) y electroimanes, la calibradora expulsará la fruta a las mesas de confección seleccionadas previamente por el operador en el PC de máquina principal (ver siguientes ilustraciones).

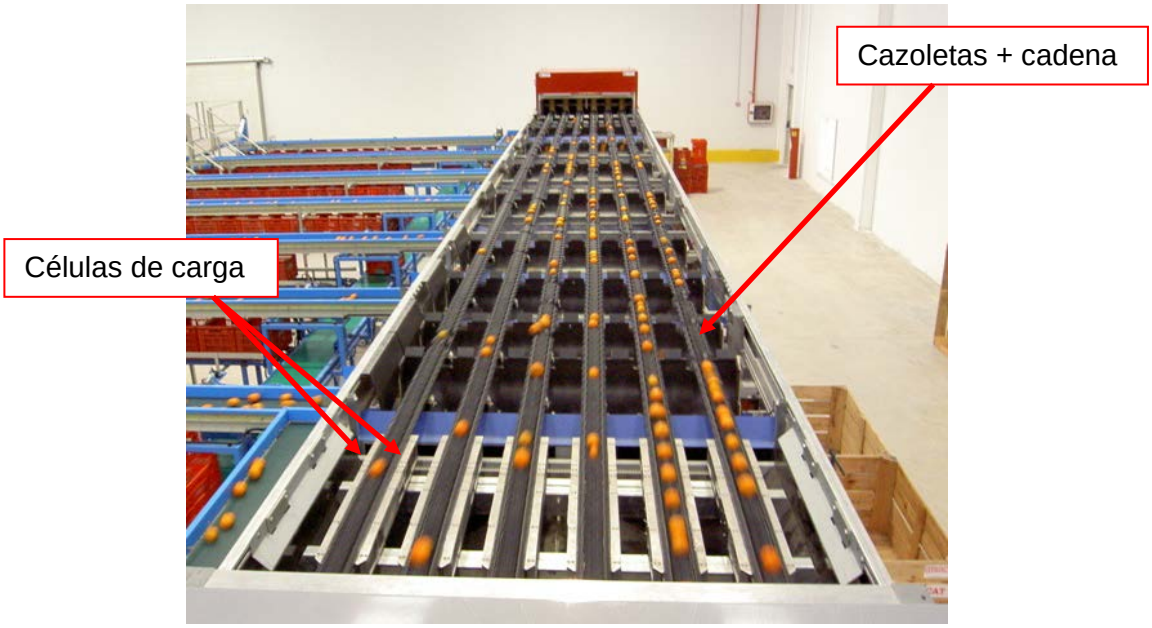


Ilustración 15

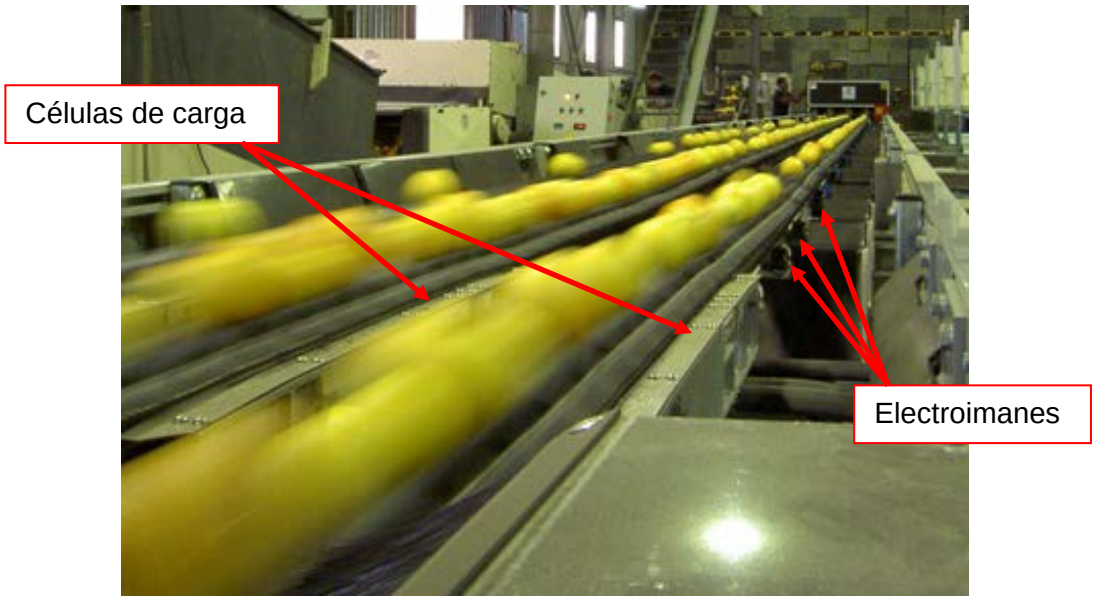


Ilustración 16

(18) <u>Transportador de cajas llenas</u> y (19) <u>Tramo de rodillos:</u>

Finalmente, las cajas confeccionadas de frutas con sus diversos tamaños calibrados, salen a través de un circuito de transportadores y tramos de rodillos donde un último operario los agrega a un palet para que cuando este se complete pueda ser almacenado en cámaras de refrigeración y/o maduración o incluso enviarse a los camiones para ser distribuido a sus correspondientes destinatarios.

Ilustración 17

3. VISIÓN ARTIFICIAL

3.1. Descripción

El sistema de visión artificial automático de calibración de fruta por diámetro debe ejecutar las siguientes tareas:

1. Adquisición de las imágenes en la línea de producción (hardware)
2. Procesamiento de la imágenes (hardware / software)
3. Gestión de los resultados (hardware / software)
4. Comunicación con la unidad principal calibradora (hardware / software)

Así pues, para la ejecución de estas tareas, el proyecto requiere de un hardware y un software que nos permita obtener óptimos resultados.

El objetivo del sistema de calibración es poder trabajar con las siguientes características:

- Velocidad de calibración: 5 unidades /segundo
- Margen de error: +-1mm
- Diámetro máximo: 79mm
- Fruta a calibrar: ciruelas y nectarinas
- Número de líneas de calibración: 4 líneas
- Área total de visión: 780 x 316mm

3.2. Hardware

3.2.1. Visión global

El módulo de visión artificial objeto del presente proyecto requiere del siguiente hardware:

- Cámaras y ópticas
- Sistema de iluminación
- Procesador industrial
- Sistema de comunicación

De este modo, el hardware mencionado tendría la siguiente configuración en máquina:

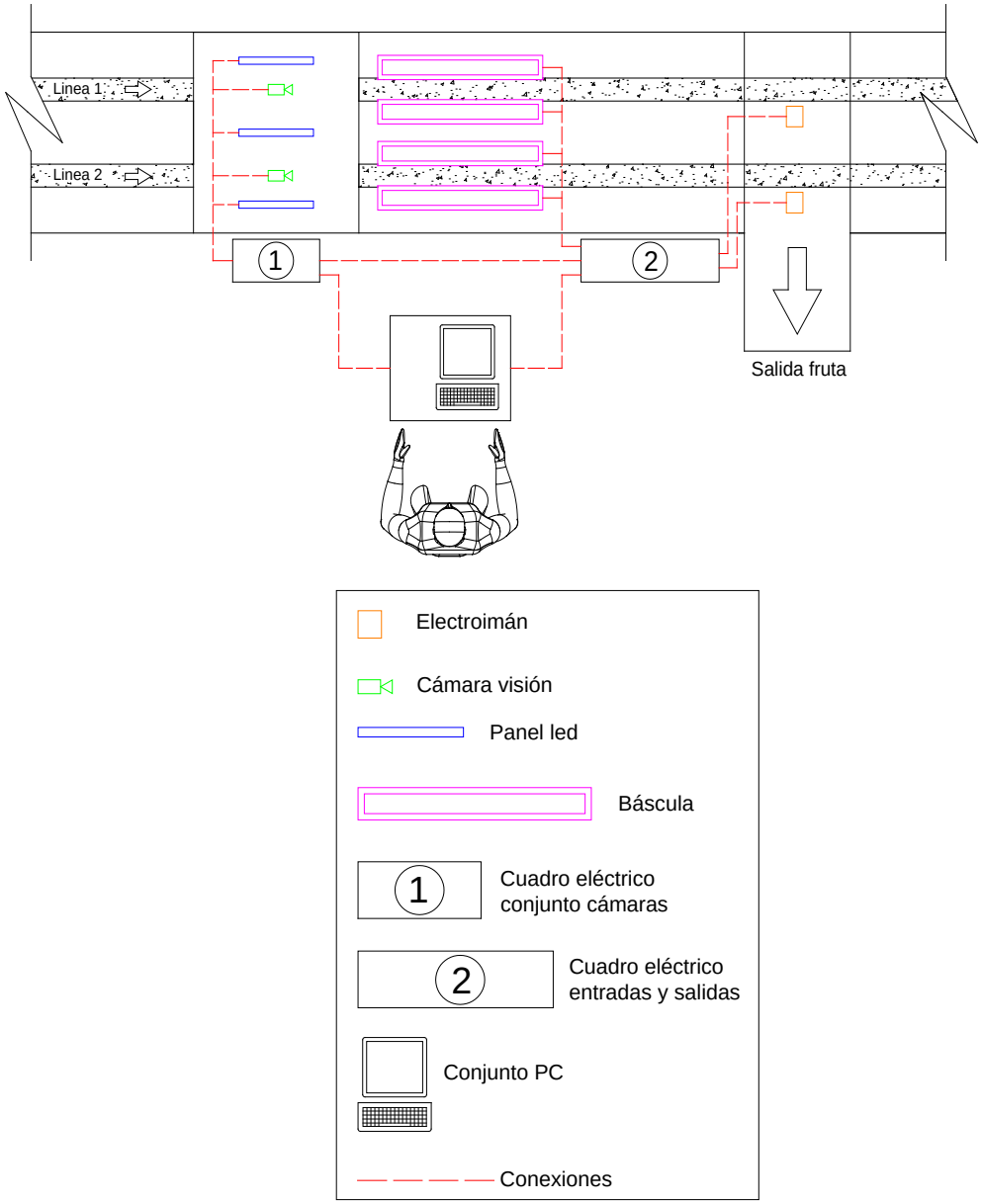


Ilustración 18

3.2.2. Cámaras y ópticas

Debido a que necesitamos trabajar con fruta lo suficientemente oscura, necesitamos un sistema de visión artificial sofisticado que sea capaz de diferenciar las piezas cuyas cazoletas (sistema de manos que transporta la fruta en la línea de producción) también son oscuras.

Actualmente, el método utilizado para la detección de diámetro de fruta oscura es utilizando cazoletas de color claro (azul cian para obtener un mayor contraste) y cámaras de color convencionales. El gran inconveniente de esta metodología es que las cazoletas se ensucian y la suciedad provocaba que las cámaras obtengan lecturas erróneas que derivan en falsos valores de diámetro.

A este inconveniente cabe añadirle que las cámaras de color convencionales son demasiado delicadas al cambio de intensidad de luz. Debemos tener en cuenta que para trabajar con color, antes es necesario hacer un mapa de colores y así discriminar (por software) el entorno a la fruta (la zona de cazoletas). Sin embargo, debido al cambio de intensidad de luz, el mapa de colores se ve negativamente afectado de modo que las lecturas obtenidas tampoco disponen de la fiabilidad requerida en un proceso industrial.

Actualmente existe tecnología que nos permite obtener óptimos resultados en fiabilidad y velocidad de operación. Así es, el presente proyecto ofrece una ventaja competitiva de gran envergadura frente a las actuales tecnologías utilizadas debido a que este propone trabajar con cámaras que detectan márgenes de longitudes de onda propias del infrarrojo IR. Gracias a la utilización de cámaras infrarrojo podemos constatar las siguientes ventajas competitivas:

- Método mucho más sencillo para el operario (no debe hacer mapa de colores)
- Longitud de onda que permite diferenciar la fruta de fondos oscuros
- Tecnología mucho más fiable (no es tan delicado al cambio de colores)
- Menor mantenimiento (no requiere calibraciones)
- Mayor velocidad y menos requerimientos de hardware (sin mapa de colores el software es mas compacto)

Así pues, Debido a la buena relación calidad – precio que requerimos para el proyecto, hemos elegido unas cámaras con las siguientes características:

- Cámaras GENIE de 1063x1024 pixel
- Monocromo (visión en infrarrojo IR)
- Conexión por TCP/IP

3.2.2.1 Resolución de las cámaras

Para obtener la resolución se debe de dividir el área que se desea observar entre el número de pixeles del sensor. En realidad la división sería entre el ancho/alto del área a observar entre el ancho/alto correspondiente del tamaño del sensor. En este caso estamos utilizando que para los 316mm tenemos el lado del sensor que tiene 1024pixeles, haciendo la división 316/1024 tendríamos una resolución de 0.308mm/pixel. Es muy arriesgado decir que tenemos esta resolución, normalmente se multiplica este valor por 2.5 para asegurarnos y no tener fallos por ruido. Entonces estaríamos diciendo que tenemos una resolución precisa de 0.308x2.5 = 0.77mm/pixel.

3.2.2.2 Tiempo de exposición de las cámaras

Para el tiempo de exposición tenemos que para las cámaras matriciales con iluminación normal o estándar (que no sea de alta potencia) se utiliza un tiempo de exposición entre el rango de 2ms a 10ms para obtener imágenes no saturadas. Normalmente siempre tomamos un valor intermedio de unos 5 ó 6 ms para asegurar y comenzamos a modificar el iris de la lente para obtener el contraste que buscamos. Además con este tiempo podemos asegurar que se tiene una imagen excelente aun cuando se tenga movimiento en el producto, como lo sería en este caso que se mueve la línea de transporte.

3.2.2.3 Elección de las ópticas

$$DISTANCIA\ FOCAL = \frac{TAMAÑO\ DEL\ SENSOR \times DISTANCIA\ AL\ OBJETO}{TAMAÑO\ DEL\ OBJETO}$$

De esta fórmula despejaremos la distancia al objeto, que es el valor que nos interesa. Tomando en cuenta que la distancia focal será 8 o 12mm (valores comerciales existentes en el mercado), que el tamaño del sensor es de 4.7616mm (Tamaño del alto del sensor) y el tamaño del objeto sería de 316, tenemos los siguientes resultados: con la óptica de 8mm se tendría una altura en la cámara de aproximadamente 530mm, mientras que con la óptica de 12mm tendríamos una altura de 795mm aproximadamente. Debemos de tomar en cuenta que entre mayor sea la distancia focal menor será la distorsión, pero la iluminación sería más tenue así que debido a que en nuestra aplicación la iluminación es crítica utilizaremos una lente con las siguientes características:

- Goyo Optical Inc.
- Óptica de alta resolución de 8mm
- Rango del iris de F1.4
- Rosca C 2/3” con tornillos de fijación

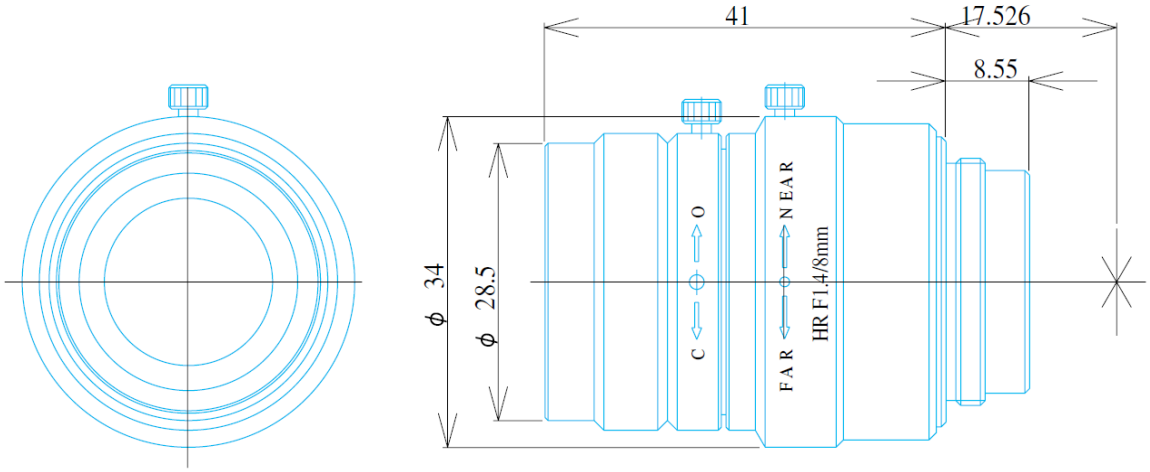


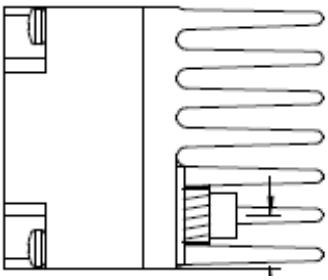
Ilustración 19

3.2.3. Sistema de iluminación

Para obtener buenos resultados en la adquisición de las imágenes, uno de los parámetros decisivos es la utilización de un buen sistema de iluminación. Actualmente existen dos sistemas de iluminación en el mercado que nos ofrecen la longitud de onda deseada en infrarrojo (850nm): iluminación por fluorescentes con un filtro añadido e iluminación por LED que ya emiten la longitud de onda IR deseada (sin la necesidad de añadir filtros). En el presente proyecto utilizaremos iluminación LED porque a pesar de que esta es más costosa no requiere filtros externos (que se ensucian y requieren mantenimiento) y la luz emitida varía muy poco con las horas de funcionamiento de estas. Este último factor es decisivo para nuestra aplicación debido a que los fluorescentes varían su emisión de luz con la temperatura u horas de funcionamiento provocando que las cámaras se descalibren y afecten a la calidad de imagen adquirida.

Características de las luces:

- Barra de LEDs de 400x25mm
- LEDs de 5mm a 12Vdc
- Tensión de alimentación: 12Vdc / 11,76W
- Método de refrigeración: disipador de potencia
- Área de iluminación: 400x16mm2
- Dimensiones: 422x24x23
- Material: aleación de aluminio
- Temperatura de trabajo: -5 a 45°C



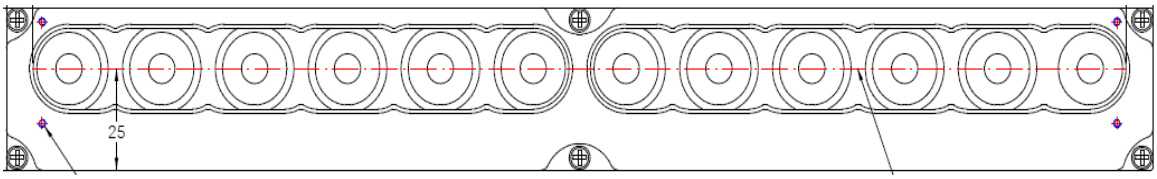


Ilustración 20

3.2.4. Sistema estroboscópico

Para alimentar las luces LED requerimos de un módulo estroboscópico cuya función es la de gestionar la intensidad de salida que alimenta las barras LED cuando recibe un pulso de entrada providente directamente del detector fotocélula de paso de cazoletas. Esta tarea la realizaremos a través del módulo estrobe comercial modelo PP520 del fabricante Gardasoft Vision Ltd (ver siguiente ilustración).

Ilustración 21

Este módulo programable permite configurar el nivel de intensidad que queremos dar a las barras LED (nivel de brillo del 0 al 150%), tiempo de cada pulso en milisegundos, y retardo en milisegundos (en nuestra aplicación no requerimos retardo), (ver siguiente ilustración).

Ilustración 22

3.2.5. Procesador industrial

El procesador o PC que requerimos para nuestra aplicación debe ser carácter industrial para disponer de alta fiabilidad y utilizar un Windows XP embeded con las aplicaciones básicas que requiere el software de control y cálculo de las cámaras.
Por eso que hemos elegido el siguiente controlador:

- Procesador Geva, 320GB HD, 2GB RAM
- 2.4GHz Dual Core
- Inputs: 8 + 2 triggers
- Outputs: 8 + 2 strobes
- Power: 12-30V / Temp: 0-45°C
- Comm: GigE, RS232, USB (x2)
- Camera Ports: GigE x2
- Mount: DIN Rail
- Storage: 120GB

Ilustración 23

3.2.6. Sistema de comunicación

Nuestro sistema, después de adquirir las imágenes y realizar los cálculos deseados, debe enviar los datos a través de un protocolo de comunicación rápido y fiable. De todos los sistemas de comunicación disponibles en el procesador industrial GEVA y el software de control de las cámaras (SHERLOCK), el más útil para nuestra aplicación es el TCP/IP, que a pesar de que no es un sistema muy fiable, nos permite trabajar a las velocidades deseadas y en la programación podemos mejorar la fiabilidad al nivel que requiere nuestra aplicación.

Los datos serán enviados al computador principal de la máquina calibradora que junto con los datos de las básculas asignará la fruta calibrada a la salida previamente configurada por el operario de la máquina.

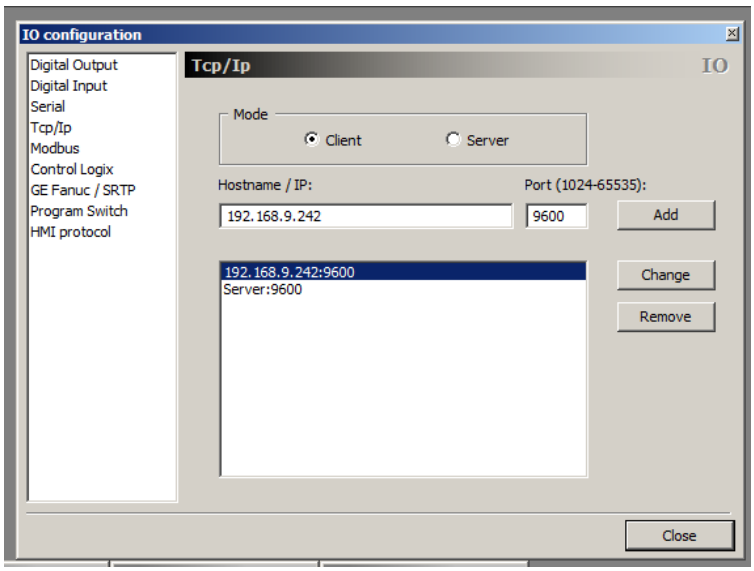


Ilustración 24

3.2.7. Alojamiento de las cámaras

Para evitar vibraciones, acceso de la suciedad a las ópticas (polvo), manipulación indebida por parte de algún operario, etc. debemos ubicar las cámaras en un habitáculo cerrado, resistente a los golpes externos y vibraciones, permitiendo un correcto ajuste y fácil instalación del sistema con una altura variable según las necesidades reales del técnico durante la instalación de las cámaras.

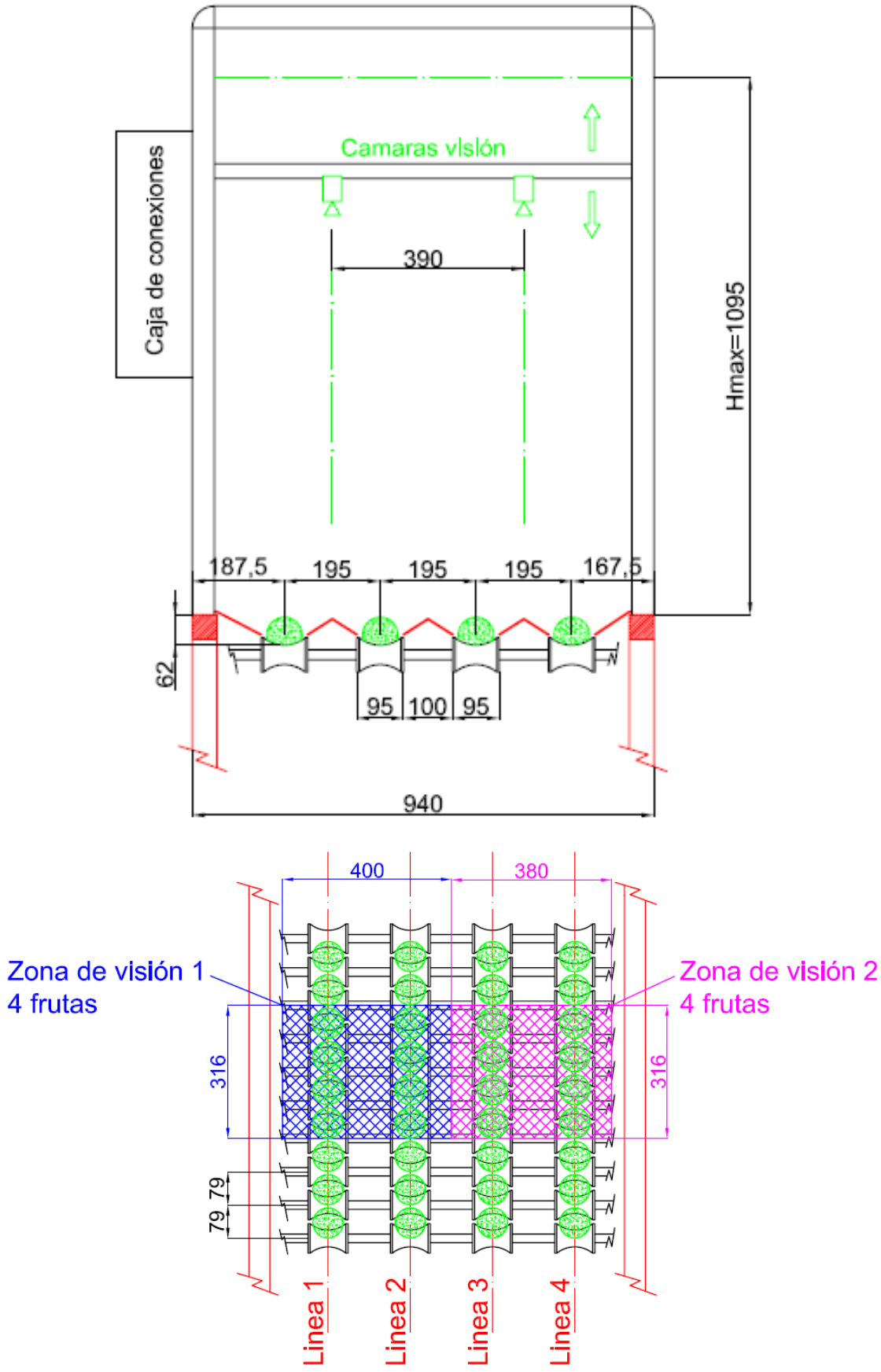


Ilustración 25

3.3. Software

Para el control de las cámaras y adquisición de las imágenes utilizaremos un software comercial llamado SHERLOCK (Sherlock V7.1.6.0) debido a que es un entorno de programación que ofrece una amplia gama de herramientas de medición y procesamiento de datos que requeriremos para nuestra aplicación.

A través del software realizaremos los siguientes procedimientos:

1. Inicio del programa
2. Adquisición de imágenes
3. Cálculo de diámetros
4. Envío de los datos al PC de la calibradora

3.3.1. Código del programa

Antes de ver y explicar el código del programa vea la siguiente ilustración:

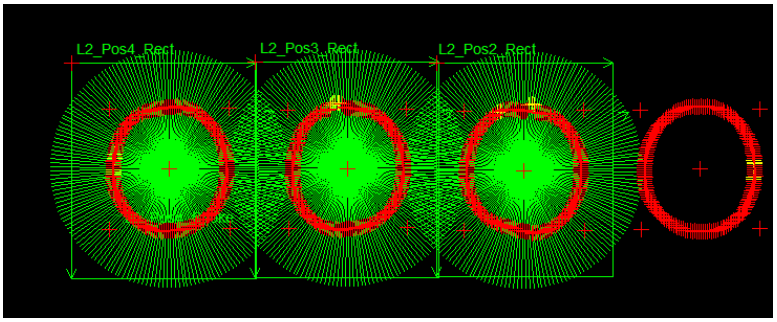
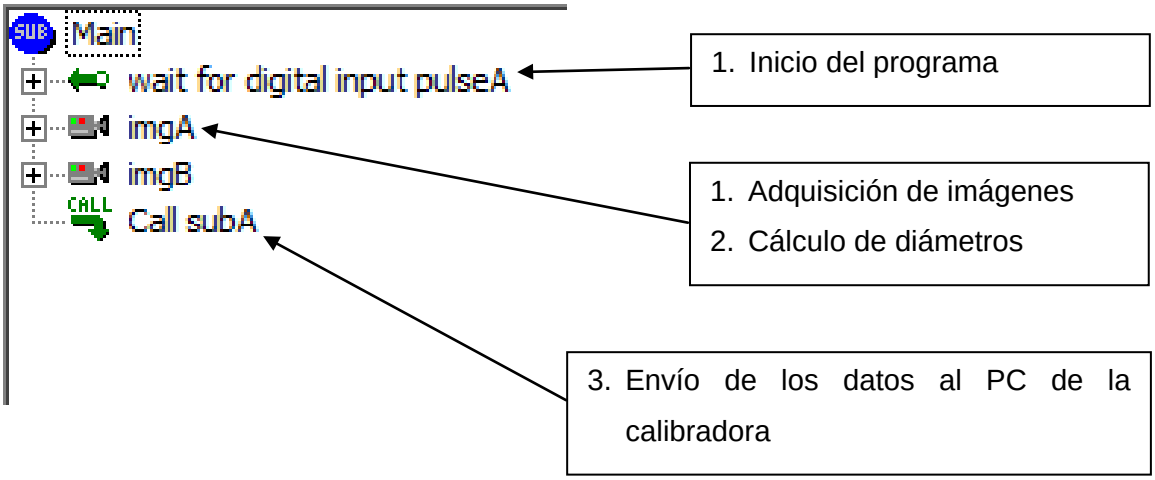


Ilustración 26

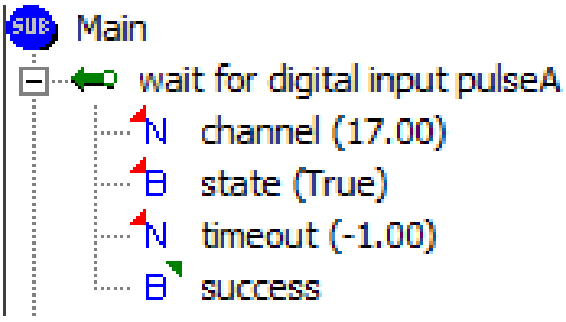
El funcionamiento del sistema de visión se basa en realizar fotos cada vez que un sensor fotocélula detecta el paso de la “cazoleta”. En el momento en que se produce dicha detección la cámara, junto con la luz estroboscópica LED, realiza una captura y ejecuta una secuencia completa del programa enviando los resultados al PLC de la calibradora. En este PC el programa principal hace una media de resultados de las 4 posiciones por donde ha recorrido la fruta. De este modo, gracias al sistema de rotación y a la captura de las 4 posiciones podemos obtener óptimos resultados debido a que captamos los 360° de la fruta.

Así pues, la estructura del programa queda del siguiente modo:



Nota: El bloque de programación “imgA” gestiona las líneas 1 y 2 de la cámara 1 y el bloque “imgB” gestiona las líneas 3 y 4 de la cámara 2.

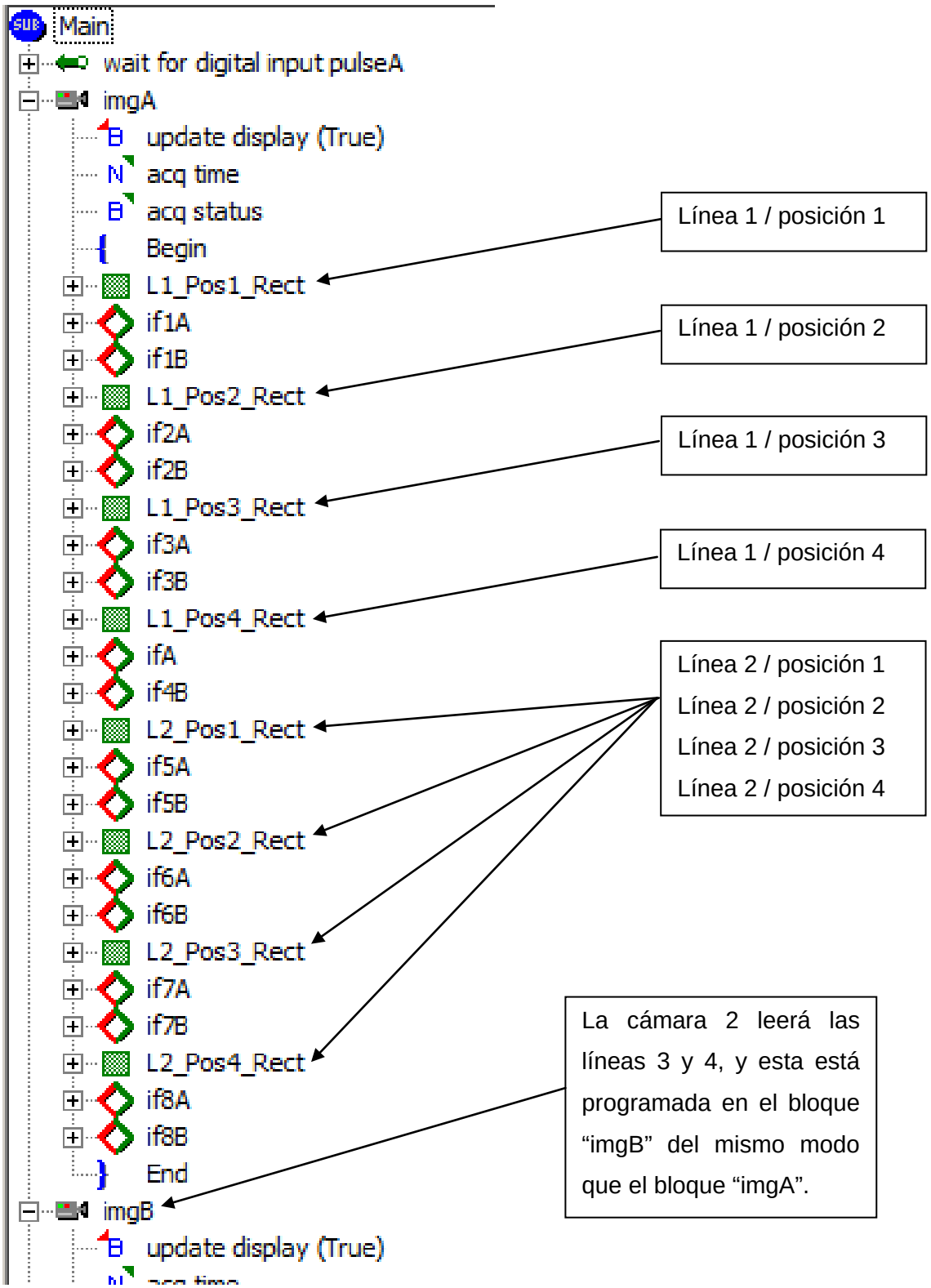
3.3.2. Inicio del programa



Tal y como hemos mencionado, el programa se ejecuta completa cada vez que el sensor fotocélula detecta el paso de las cazoletas. En este caso, tenemos configurado una entrada digital de la tarjeta física electrónica de entradas y salidas (digital input 17).

3.3.3. Adquisición de imágenes

Tal y como hemos mencionado anteriormente, cada detección de paso de cazoletas tomamos y gestionamos 4 imágenes por cada línea.


Main
wait for digital input pulseA
imgA
update display (True)
acq time
acq status
Begin
L1_Pos1_Rect
if1A
if1B
L1_Pos2_Rect
if2A
if2B
L1_Pos3_Rect
if3A
if3B
L1_Pos4_Rect
ifA
if4B
L2_Pos1_Rect
if5A
if5B
L2_Pos2_Rect
if6A
if6B
L2_Pos3_Rect
if7A
if7B
L2_Pos4_Rect
if8A
if8B
End
imgB
update display (True)
Línea 1 / posición 1
Línea 1 / posición 2
Línea 1 / posición 3
Línea 1 / posición 4
Línea 2 / posición 1
Línea 2 / posición 2
Línea 2 / posición 3
Línea 2 / posición 4
La cámara 2 leerá las líneas 3 y 4, y esta está programada en el bloque “imgB” del mismo modo que el bloque “imgA”.

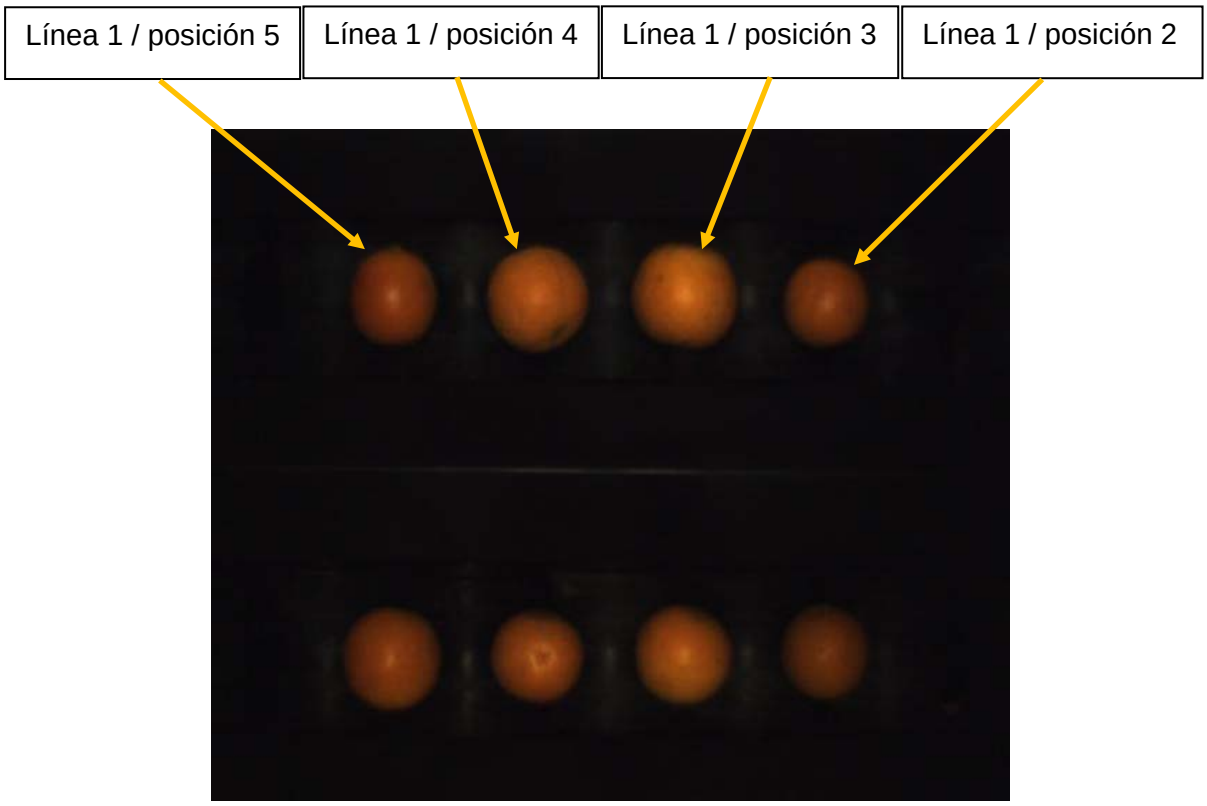
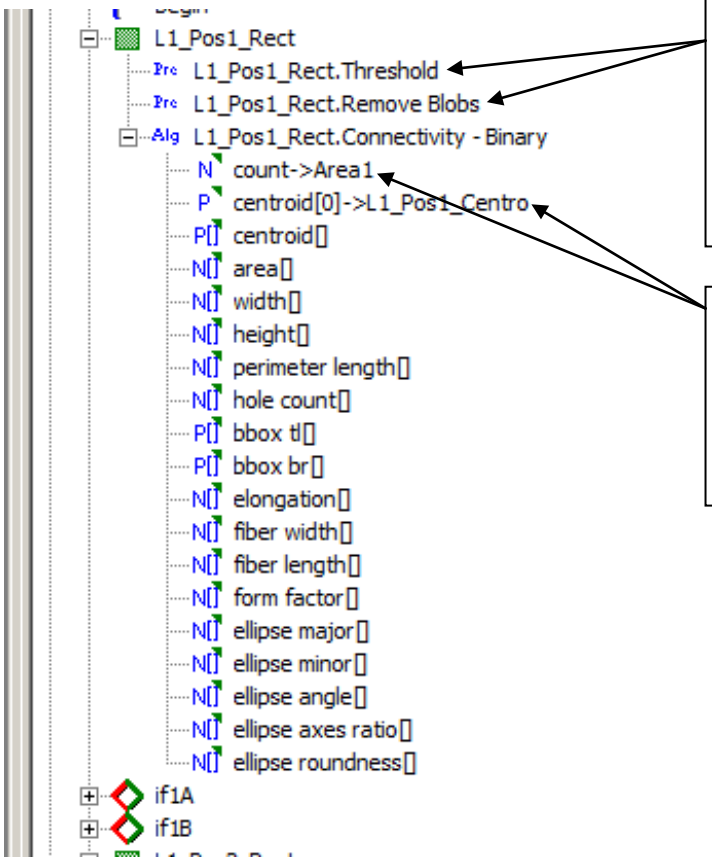


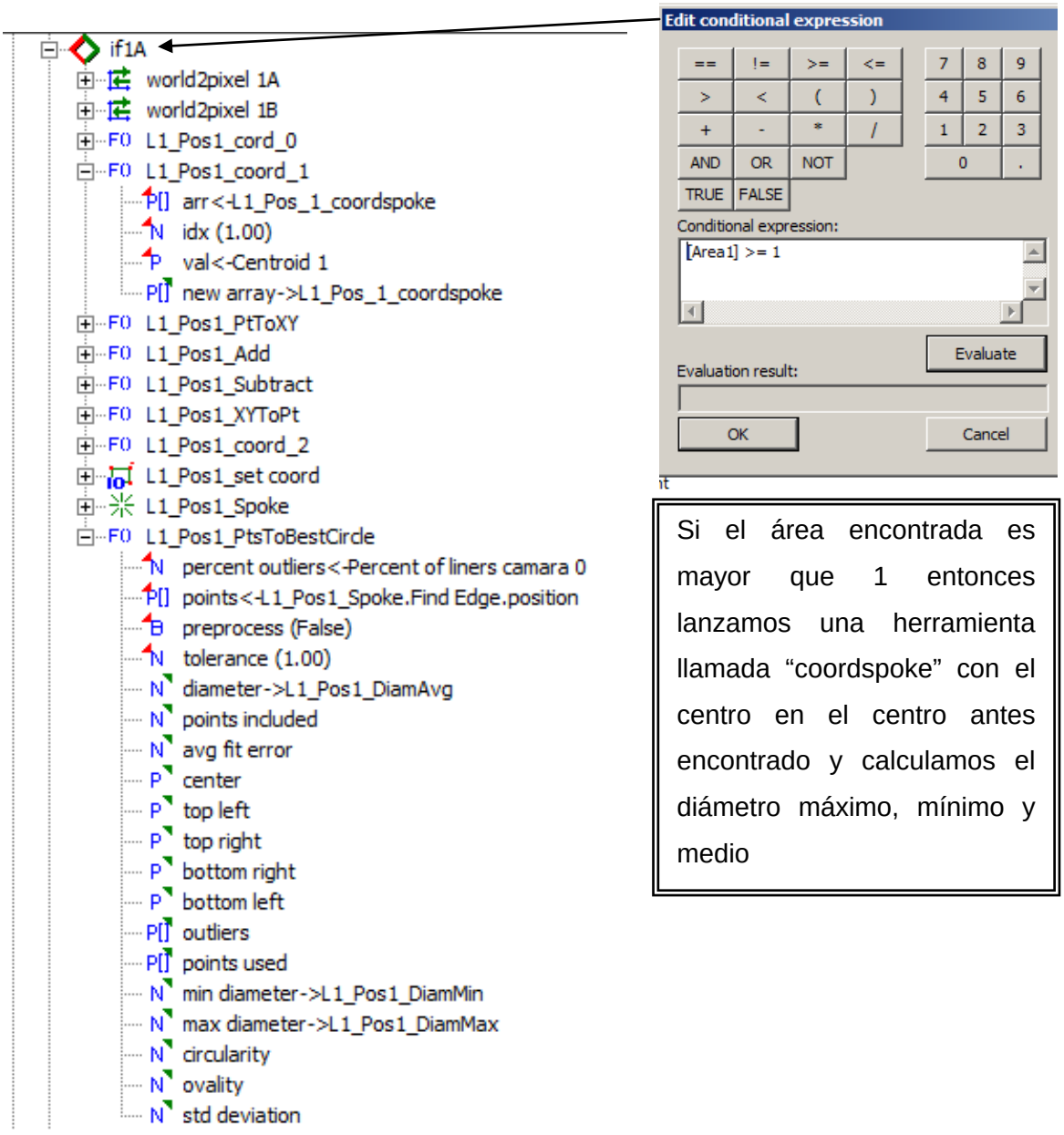
Ilustración 27

Línea X / Posición Y



Delimitamos el área de visión de “Linea X / Posición Y) y eliminamos ruido (posibles manchas y/o falsas pequeñas áreas), generamos filtros (threshold)

Dentro del área delimitada comprobamos si detectamos un área (de una fruta) y si es así buscamos su centro



Si el área encontrada es mayor que 1 entonces lanzamos una herramienta llamada “coordspoke” con el centro en el centro antes encontrado y calculamos el diámetro máximo, mínimo y medio

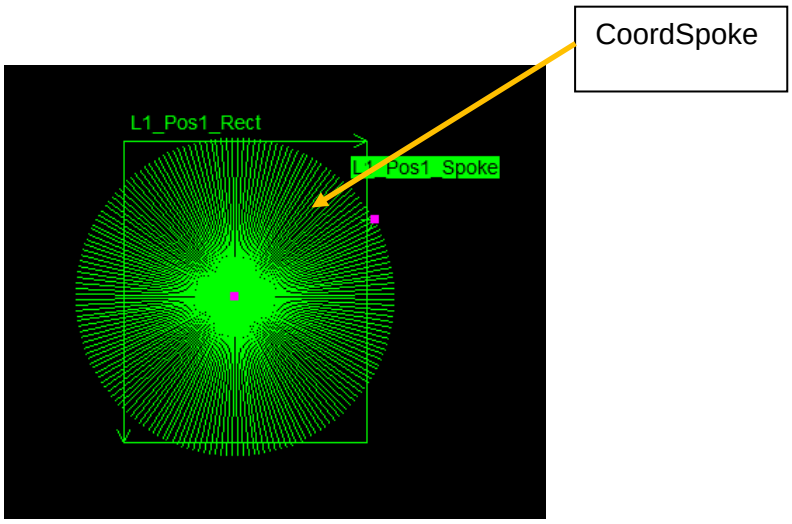


Ilustración 28

En la siguiente ilustración podemos ver el resultado de las herramientas utilizadas en el código del programa.

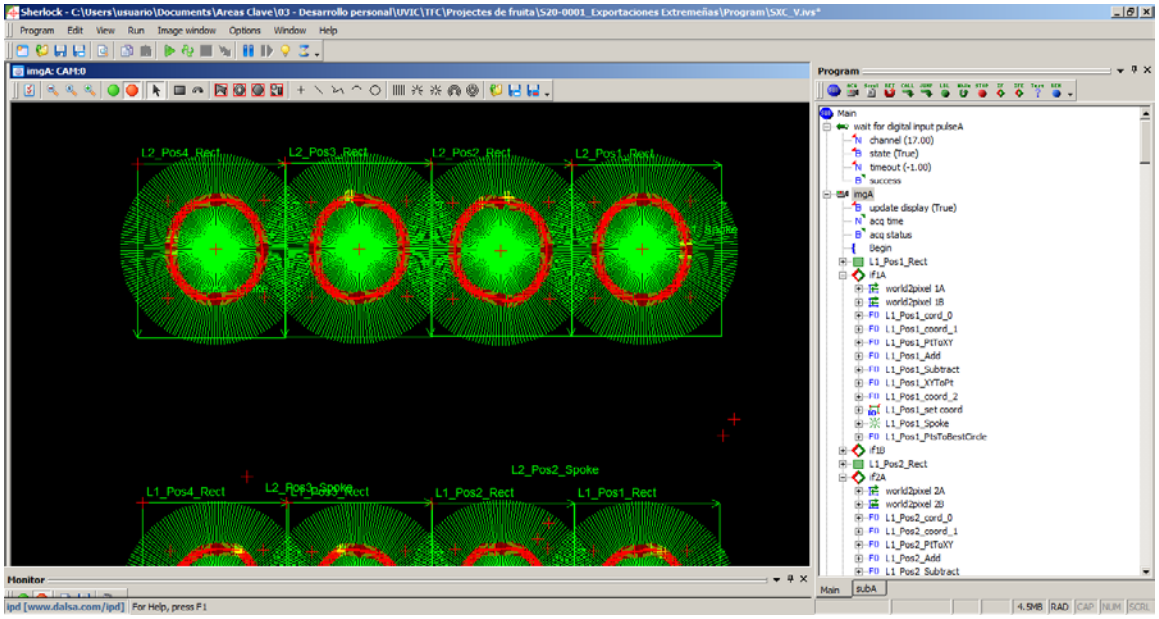


Ilustración 29

En las siguientes ilustraciones podemos ver la fruta después de aplicar los respectivos filtros aplicados en la herramienta de “threshold”.

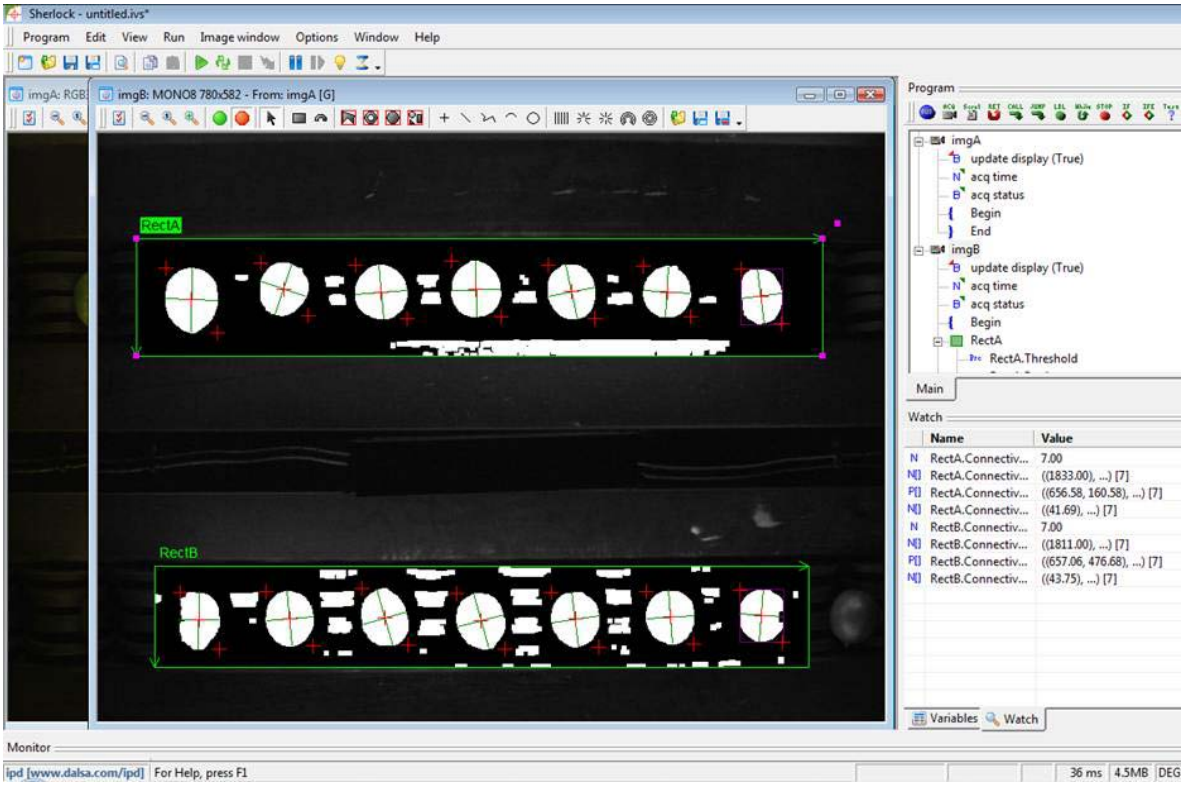


Ilustración 30

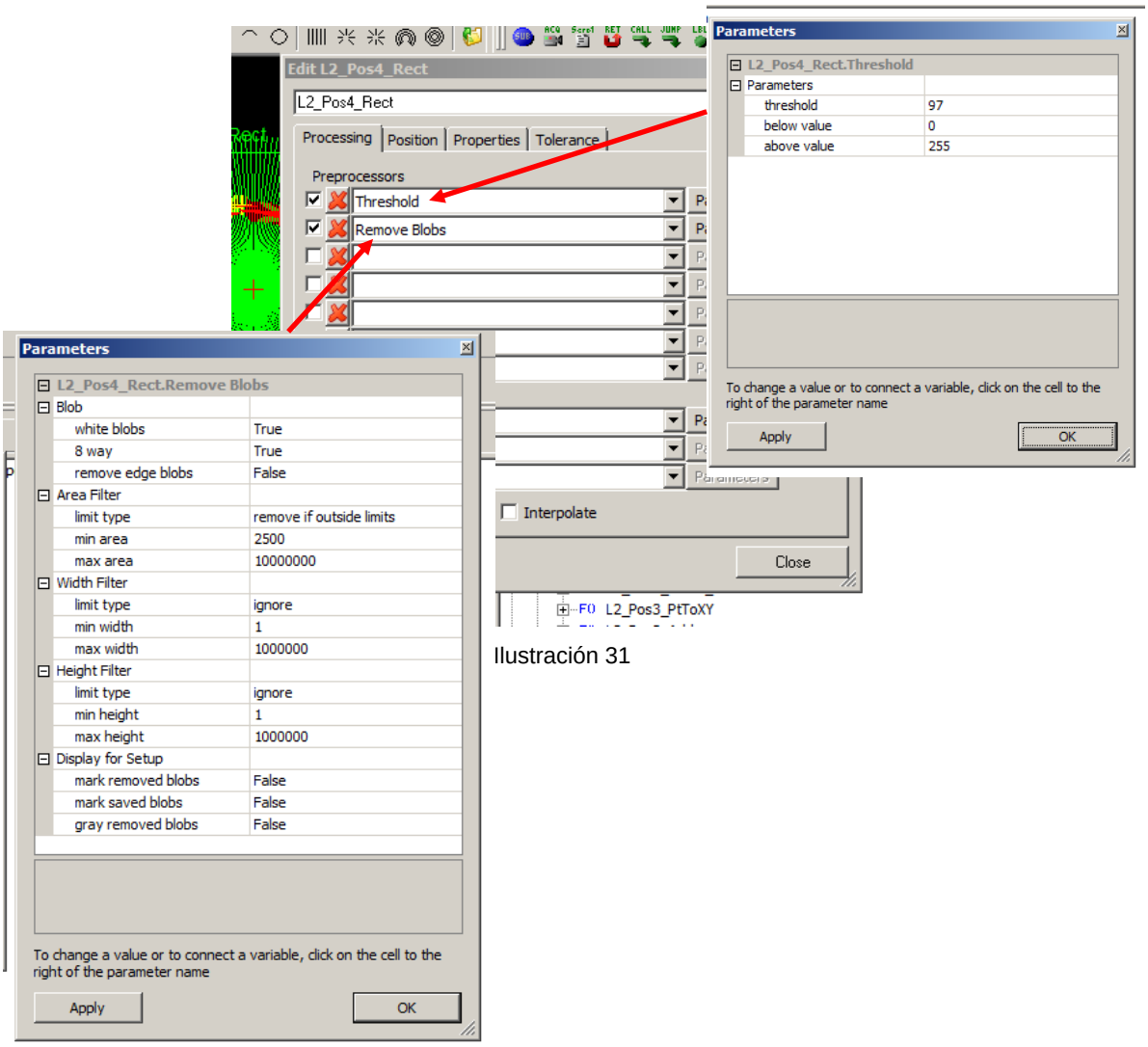
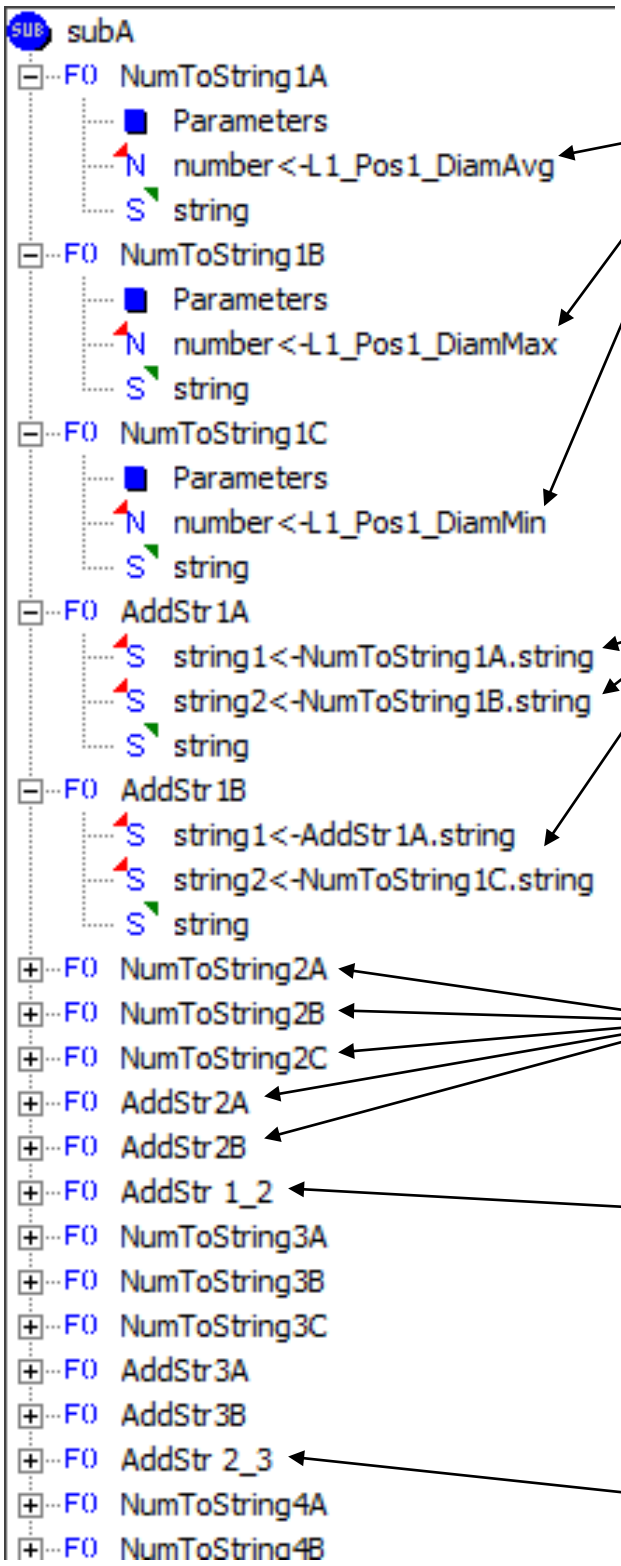


Ilustración 31

3.3.4. Envío de los datos al PC de la calibradora

Como ya hemos descrito anteriormente, cada detección de paso de cazoleta realizamos la captura y sus respectivos cálculos de 16 imágenes (4 posiciones x 4 líneas). Por lo tanto necesitaremos enviar una secuencia en formato “String” que contenga 16 x 3 = 48 resultados (Diámetro Máximo, Diámetro Mínimo y Diámetro Medio de cada posición).

Para realizar esta función convertiremos los resultados de “Integer” a “String”, los sumaremos a una única secuencia de “String” y por último enviaremos este único “String” a través de comunicación Ethernet teniendo configurado el PC de las cámaras como “server” y el PC de la calibradora como “Client”.



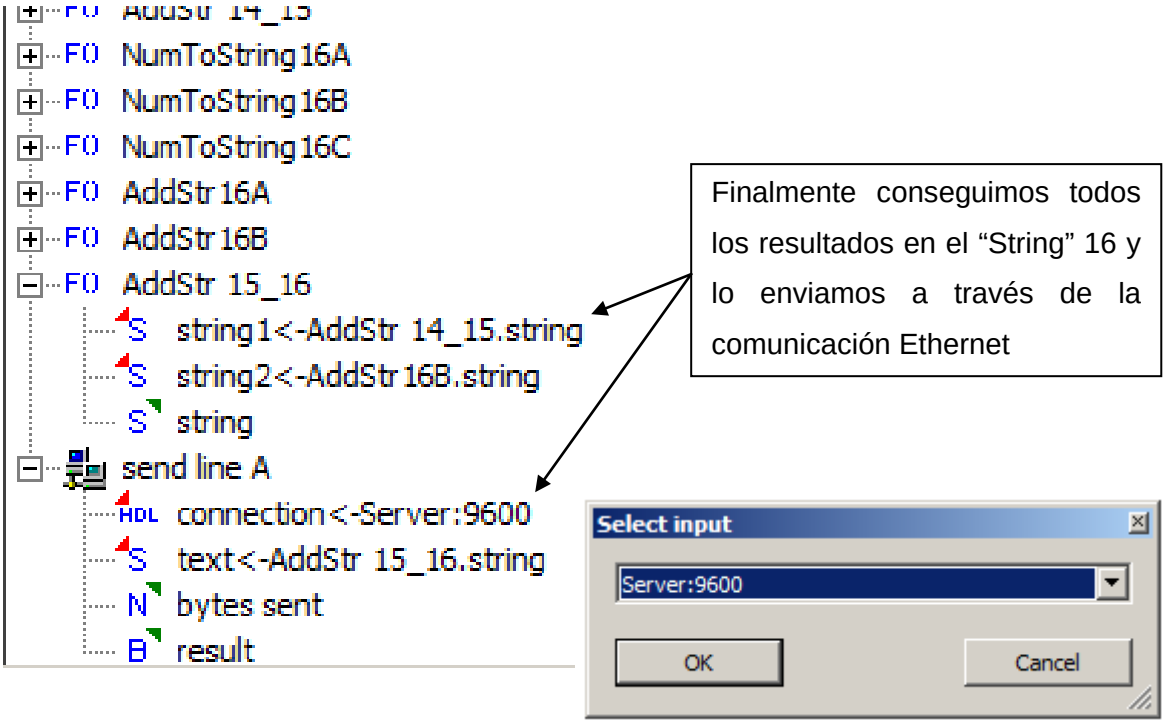
Convertimos las variables de diámetros calculados a formato "String"

Generamos la secuencia: Diámetro Medio Línea 1 Posición 1 + Diámetro Máximo Línea 1 Posición 1 + Diámetro Mínimo Línea 1 Posición 1

Ejecutamos la misma secuencia con la línea 2

Sumamos en la secuencia el "String" de la línea 1 con el "String" de la línea 2 y el resultado se almacena en el "String" de la línea 2

Línea 2 + Línea 3 y el "String" de la línea 3 almacena en esta posición todos los resultados



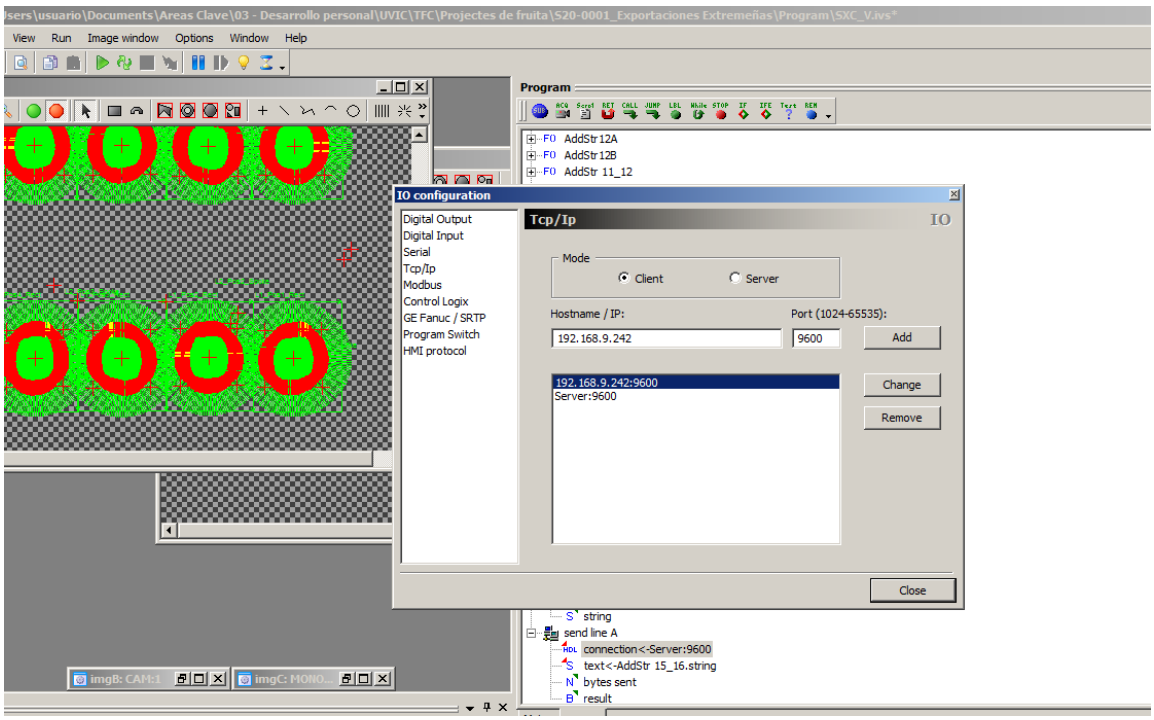


Ilustración 32

3.3.5. Resumen de programación

Como la imagen que tenemos es en blanco y negro (visión monocromo infrarroja IR), primero aplico una serie de filtros para resaltar solamente los diámetros que nos interesan y eliminar ruido que recibamos de la línea de producción. Ya teniendo una imagen con los diámetros definidos se procede a realizar la búsqueda de los conjuntos de pixeles y sacar las medidas que nos interesan, área, tamaño de elipse mayor y coordenadas en el área de interés para identificarlos.

Como se puede observar el software nos entrega todos estos valores y solo nos hace falta pasar estos valores a Strings y enviar toda la secuencia de resultados al ordenador principal de la calibradora que junto con los parámetros configurados por pantalla por el operador, la calibradora asignará la fruta a la salida requerida.

El dato que nos interesa es que toda la secuencia del programa se ejecuta en un tiempo de procesamiento de 36ms, hay que tomar en cuenta que el equipo donde se ha realizado esta prueba tiene un procesador de un solo núcleo y el equipo que se incluye en este proyecto (procesador GEVA descrito en el apartado 3.1.3.) es de doble núcleo de 2.4GHz y 2Gb de memoria RAM.

Considerando que necesitamos trabajar a una velocidad de 5 unidades por segundo, tenemos lo siguiente:

$$\frac{1}{5\,unidades/segundo} = 0{,}2segundos = 200mseg. < (36mseg. + Vcalib)$$

- Vcalib.: Velocidad de procesamiento de datos del ordenador principal de la calibradora

De este modo se confirma que no tendremos inconvenientes de velocidad para la gestión de las imágenes a la velocidad máxima de calibración.

3.3.6. Calibración de las cámaras

Para la realización de la correcta medición necesitamos calibrar las cámaras de modo que estas sepan convertir las distancias que visualizan en unidades métricas (mm).

Para ello, necesitamos diseñar una plantilla que abarque toda el área de visión de cada una de las líneas (ver siguiente ilustración) y la situemos bajo la cámara para realizar la calibración.

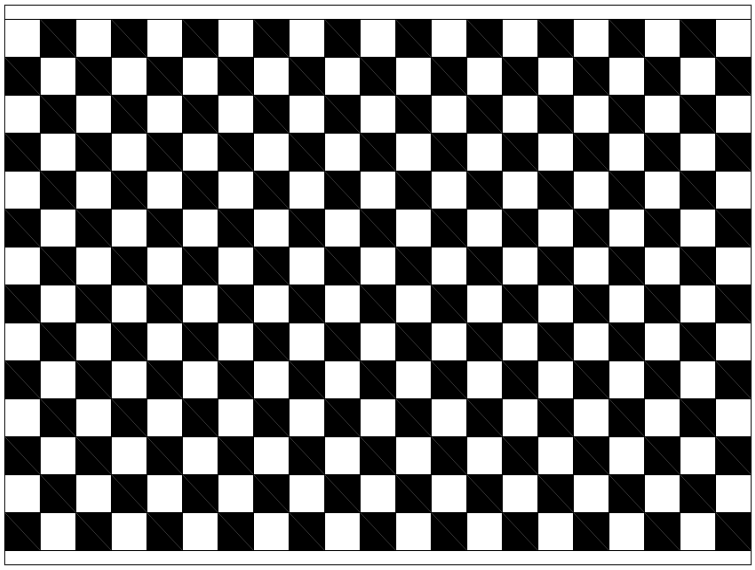


Ilustración 33

En nuestra aplicación, el entorno de programación Sherlock dispone de una función de calibración de modo que solo debemos indicar el tamaño de las cuadrículas (19,3mm) y el software realiza una auto calibración.

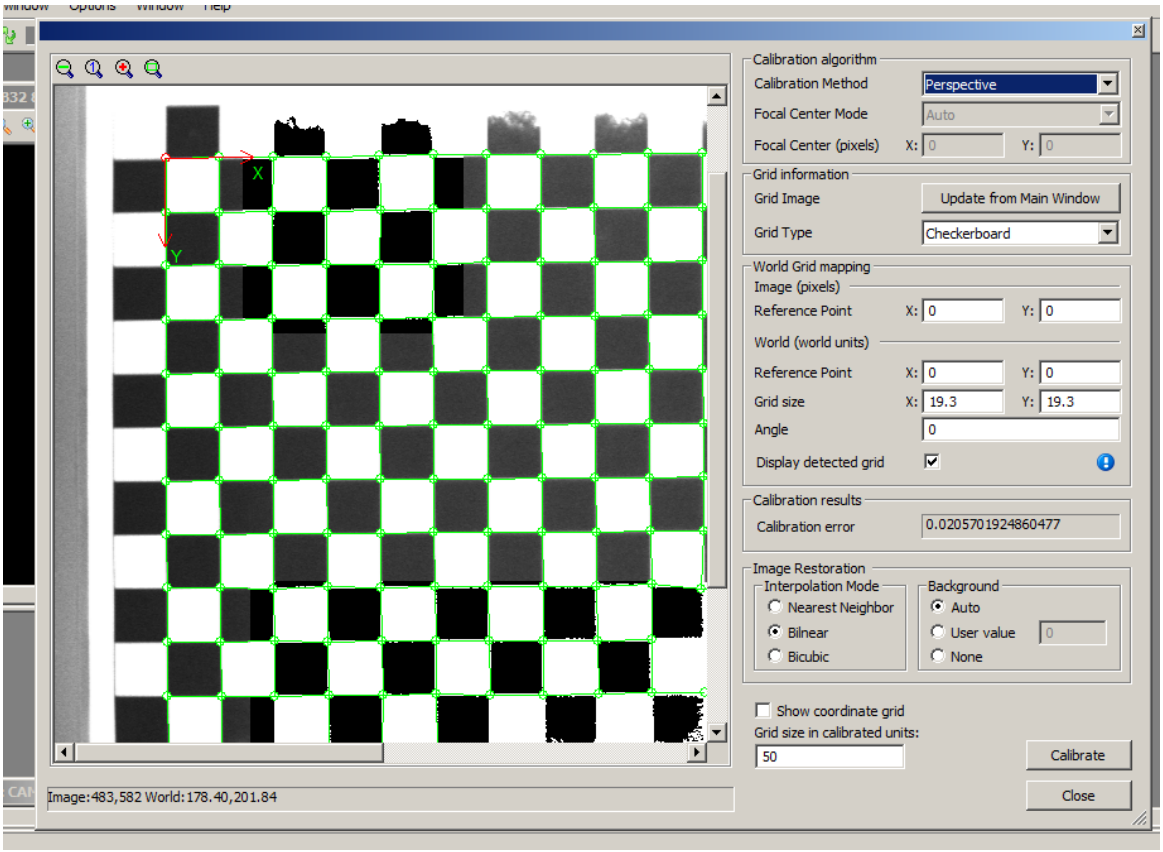


Ilustración 34

Dicha calibración debemos realizarla cuando las cámaras sean físicamente instaladas y fijadas en la máquina y solo será necesario realizar otra calibración en el caso en que estas se hayan movido.

4. CALIBRADORA

4.1. Descripción

Si nos fijamos en la siguiente ilustración, la unidad de calibración dispone de una etapa de alimentación de la fruta (módulos 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 y 12), etapa de visión artificial (módulos 13 y 14), etapa de calibración (módulos 15 y 17) y por último la etapa de salida y clasificación (módulos 16, 18 y 19).

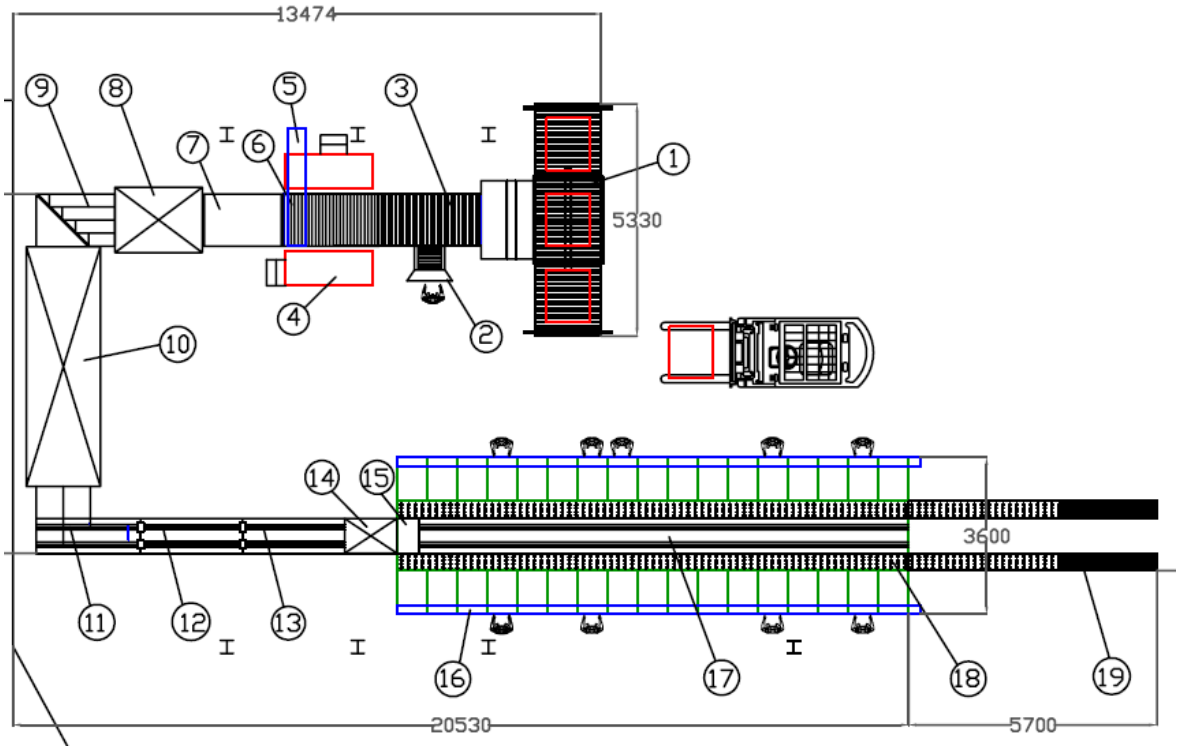


Ilustración 35

En este apartado describiremos la etapa de calibración, cuya función es la de recibir los datos del módulo de visión artificial, procesarlos y por último clasificar las piezas según los resultados.

Cabe mencionar que el punto que más nos interesa es la recepción de los datos del módulo de visión artificial, el procesamiento de estos y por último el cálculo de la salida asignada, es por este motivo que será donde dedicaremos gran parte del contenido del presente proyecto.

4.2. Hardware

4.2.1. Visión global

Si recordamos en los anteriores apartados, la distribución del hardware de la calibradora resulta del siguiente modo:

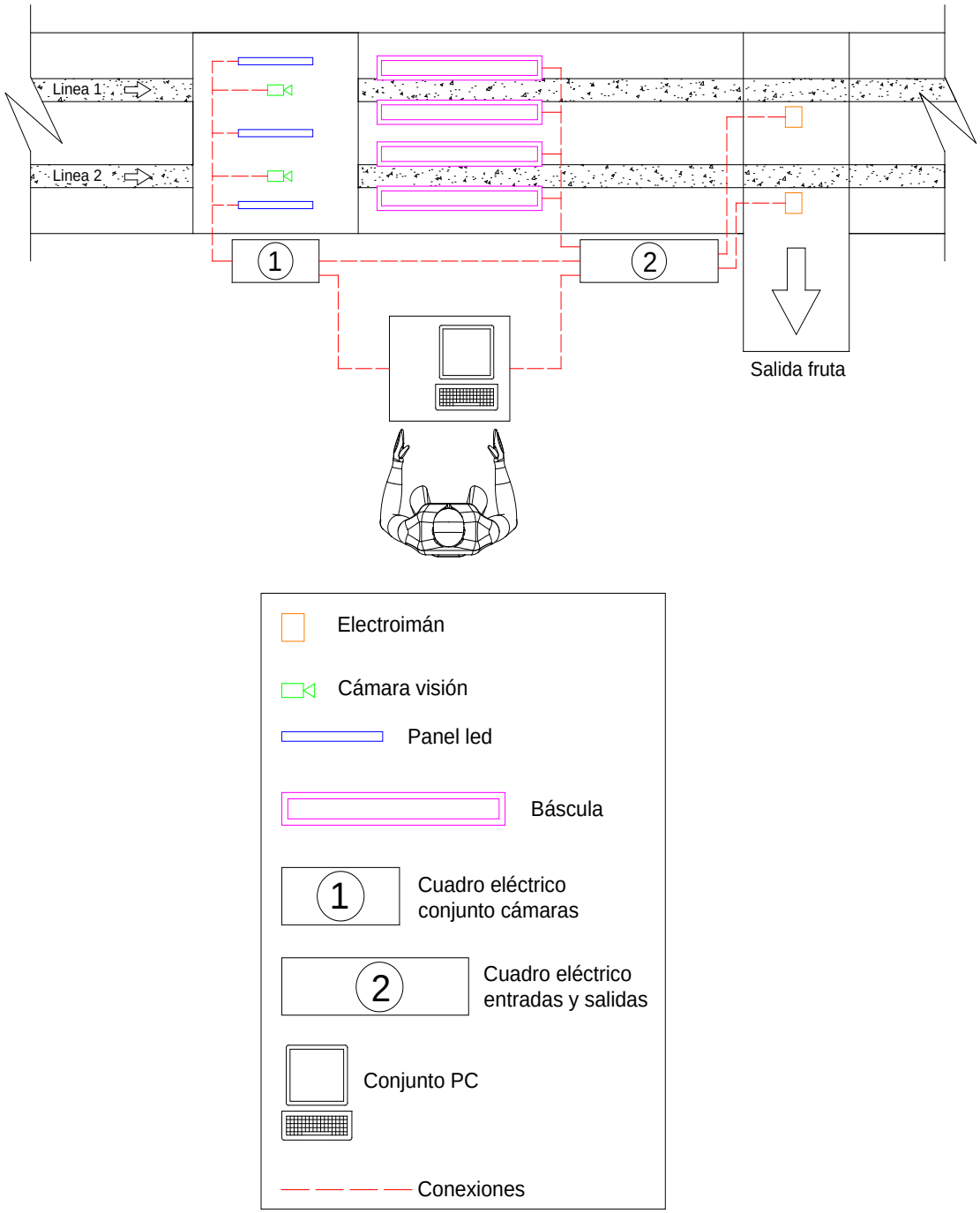


Ilustración 34

Electroimanes:

Mediante unos electroimanes de 24Vdc activados directamente de la salida del PLC conseguimos volcar las “cazoletas” de modo que la fruta sale a las mesas de confección a través de unos tapices inferiores.

Ilustración 36

Tal y como podemos apreciar en la presente ilustración, la bobina de 24Vdc es excitada a través del módulo de salidas del PLC y esta levanta la palanca naranja que vuelca las cazoletas que contienen la fruta que requerimos expulsar por esa salida.

Básculas:

A pesar de que el sistema propuesto en el presente proyecto basa su funcionamiento en la visión artificial, opcionalmente la calibradora dispone de dos módulos de células de carga por cada línea (uno a cada costado del paso de la cazoleta), cuya media entre ambas es la que nos resulta el peso dinámico total de cada fruta (ver siguientes ilustraciones).

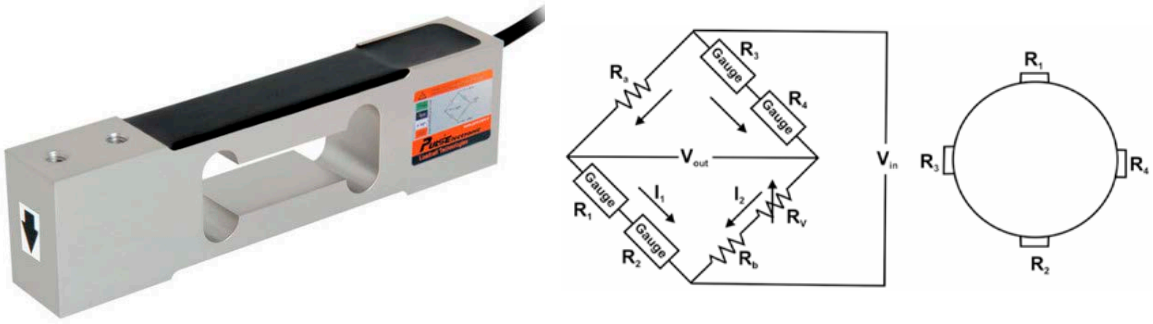


Ilustración 37

Cuadro eléctrico cámaras:

El cuadro eléctrico de las cámaras aloja el módulo estrobe que alimenta las luces LED generando pulsos cada vez que recibe la señal de la fotocélula que detecta el paso de cada cazoleta. Este también alimenta las cámaras y la alimentación de 24Vdc la recibe del cuadro eléctrico general.

Cuadro eléctrico entradas y salidas

Este cuadro eléctrico aloja los módulos de entradas y salidas del PLC que interactúan entre los elementos expuestos en la máquina calibradora y el procesador industrial PLC. Debido a la necesidad de utilizar electrónica flexible, de alta gama, robusta, que trabaje a altas velocidades y que permita un elevado nivel de programación, he elegido la electrónica que ofrece el fabricante de PLC's, "Bernecker and Rainer, B&R"

1. **Módulo de comunicación X2X:**

Para la comunicación entre los módulos de entradas y salidas y el PLC, requerimos de un módulo de comunicación propia de B&R (comunicación X2X, RS232). Este módulo es el "X20 PS 9400" (ver siguiente ilustración).

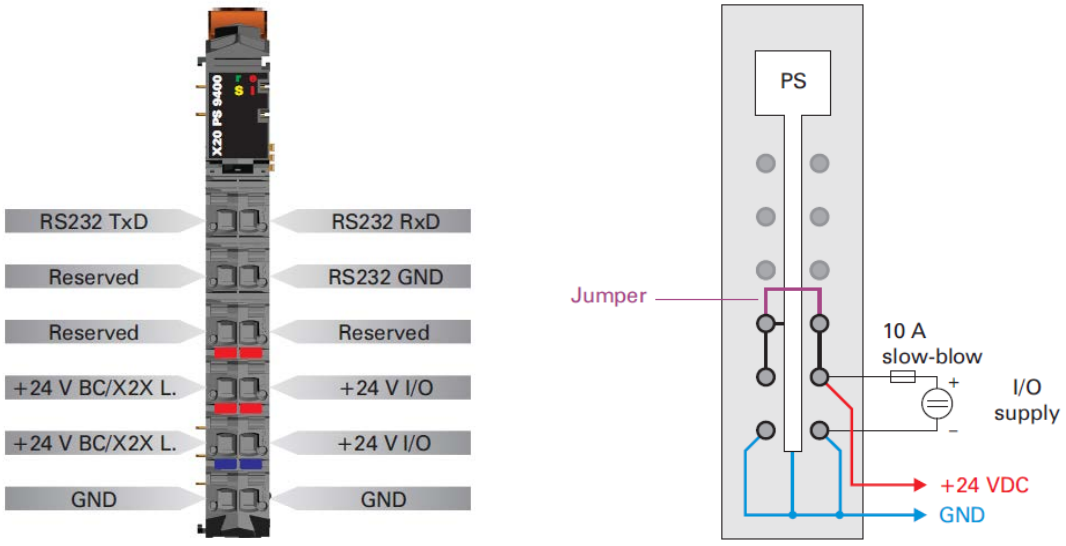


Ilustración 38

2. **Módulos de entradas digitales:**

Para la lectura de las fotocélulas de detección de paso de cazoleta utilizamos un módulo de 4 entradas digitales "X20 DI 4371" (ver siguientes ilustraciones).

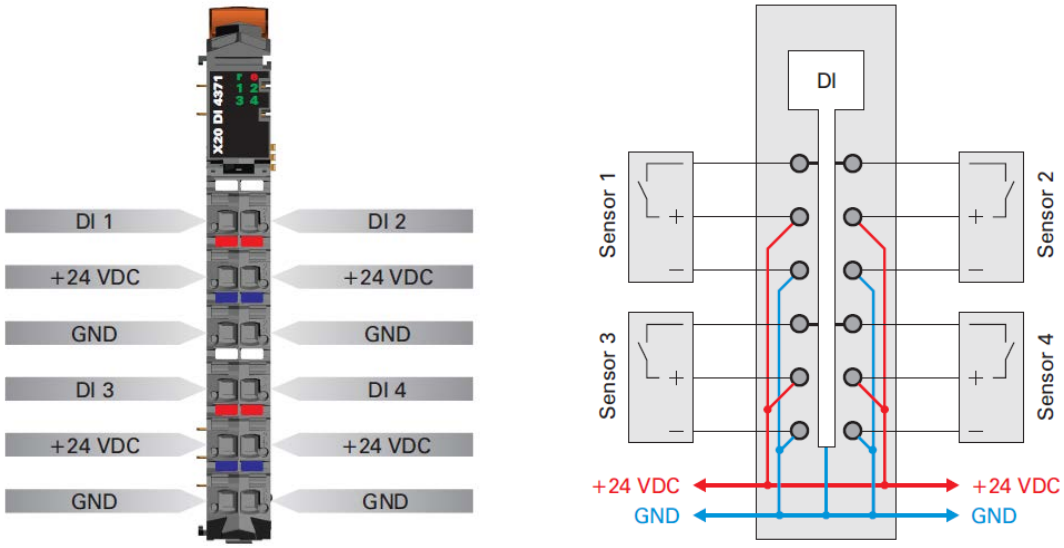


Ilustración 39

3. Módulos de salidas analógicas:

Mediante los módulos de salidas analógicas obtendremos una señal de 4-20mA que utilizaremos para el control de los variadores de frecuencia “Altivar 31” de la marca Telemecanique (señal de entrada entre “COM” y “AI3”). Estos variadores de frecuencia serán los que controlen las velocidades de la calibradora, tramos de rodillos, cepilladora, alineador, pre-alineador, etc. mencionados en el apartado “2.2. Proceso global calibración”. El módulo de salida analógica es el “X20 AO 4022” (ver siguientes ilustraciones).

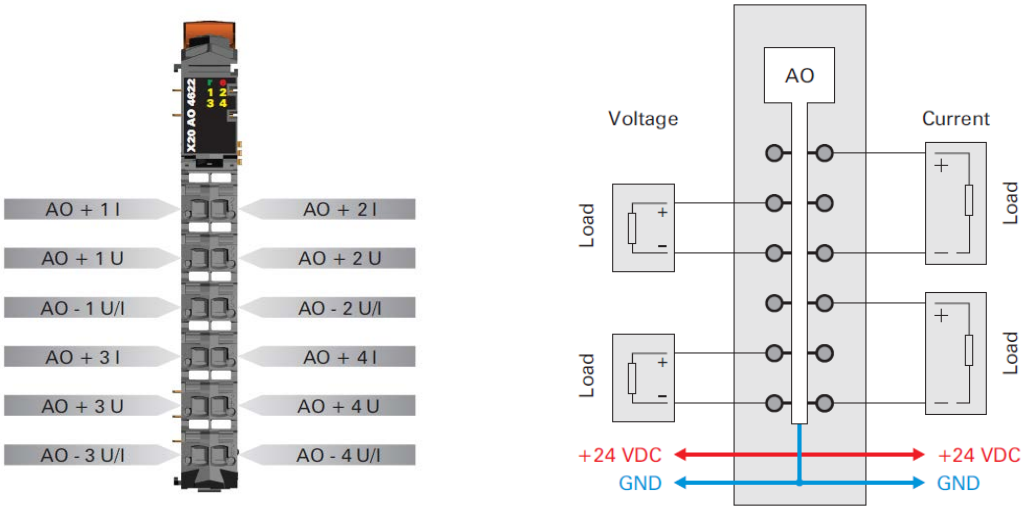


Ilustración 40

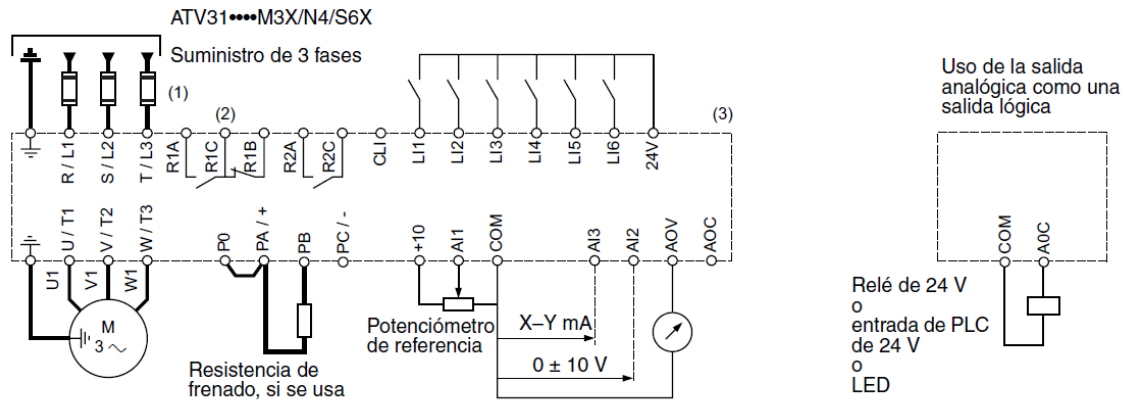


Ilustración 41

4. Módulos de entradas analógicas:

B&R dispone de unos módulos especialmente diseñados para la lectura de células de carga ideal para nuestra aplicación en la conexión de las básculas mencionadas anteriormente. Cada módulo permite la conexión de dos células de carga que por software podemos configurar que el resultado obtenido sea la media de la lectura de ambas entradas.

El módulo mencionado es el “X20 AI 1744” (ver siguientes ilustraciones).

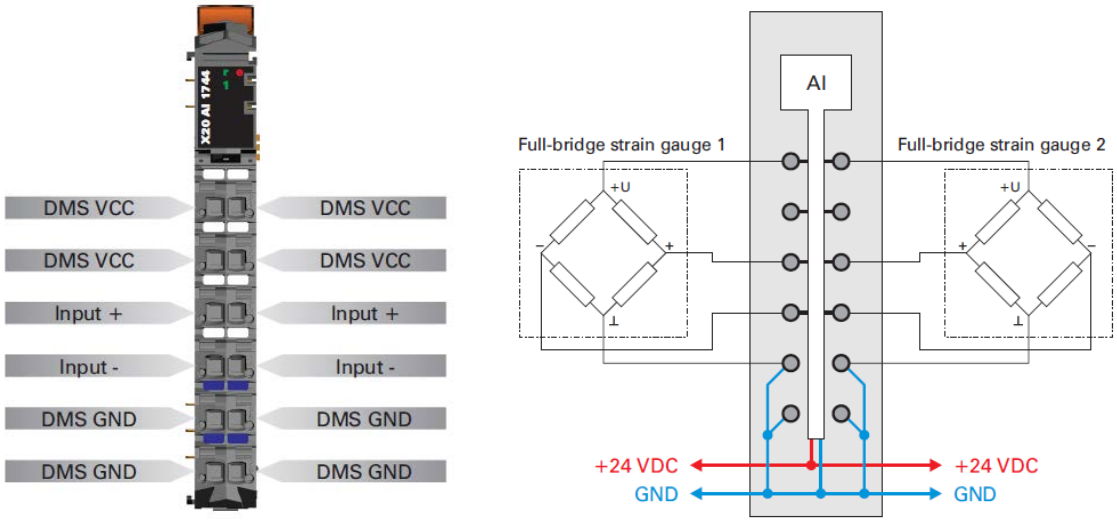


Ilustración 42

5. Módulo de alimentación:

Debido al consumo de los electroimanes, requerimos del empleo de unos módulos diseñados por B&R que proporcionan más capacidad de carga (mA) en la salidas digitales. Este módulo es el “X20 PS 2100” (ver siguiente ilustración)

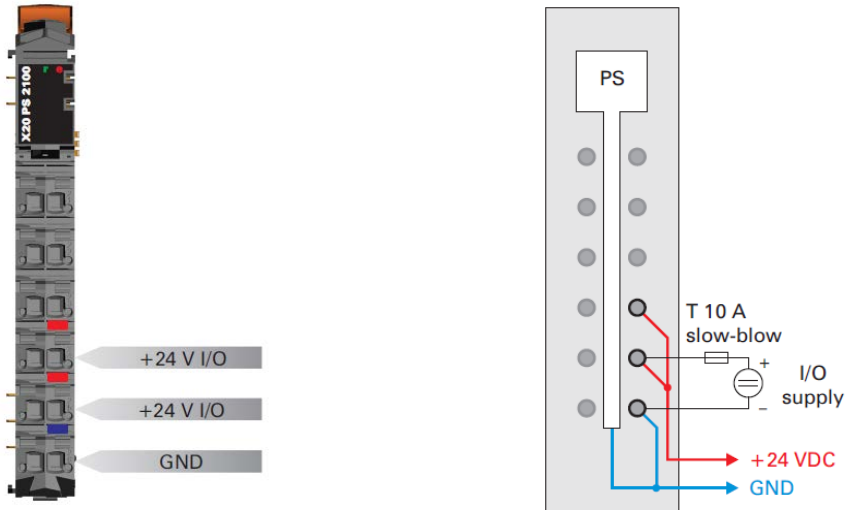


Ilustración 43

6. Módulos de salidas digitales:

Los módulos de salidas digitales nos proporcionarán la alimentación de 24Vdc de los electroimanes para la activación de estos en la salida deseada de la fruta. El módulo que utilizaremos es el "X20 DO 8332" (ver siguientes ilustraciones).

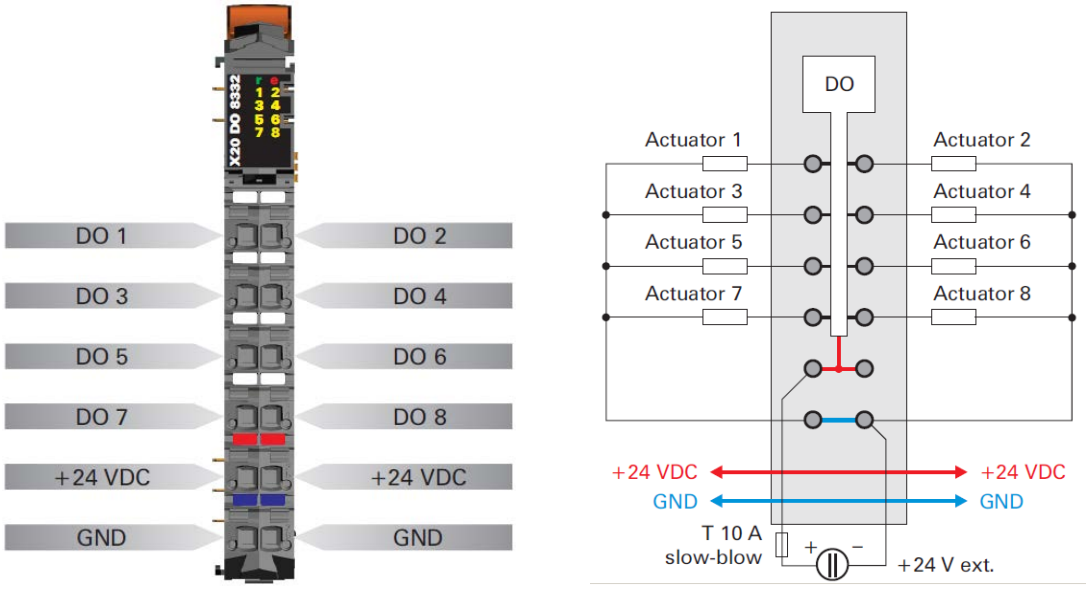


Ilustración 44

7. Otros elementos:

El cuadro eléctrico de entradas y salidas también dispondrá de una Fuente de alimentación de 24Vdc que alimentará la electrónica de toda la calibradora (PLC, iluminación y cámaras, módulos de entradas y salidas, etc.), magnetotérmicos, bornes de conexión, etc.

8. Configuración de los módulos de entrada y salida de PLC

Según lo expuesto anteriormente, los módulos de entradas y salidas quedarían instalados con la siguiente configuración (ver Anexo 1, esquemas eléctricos)

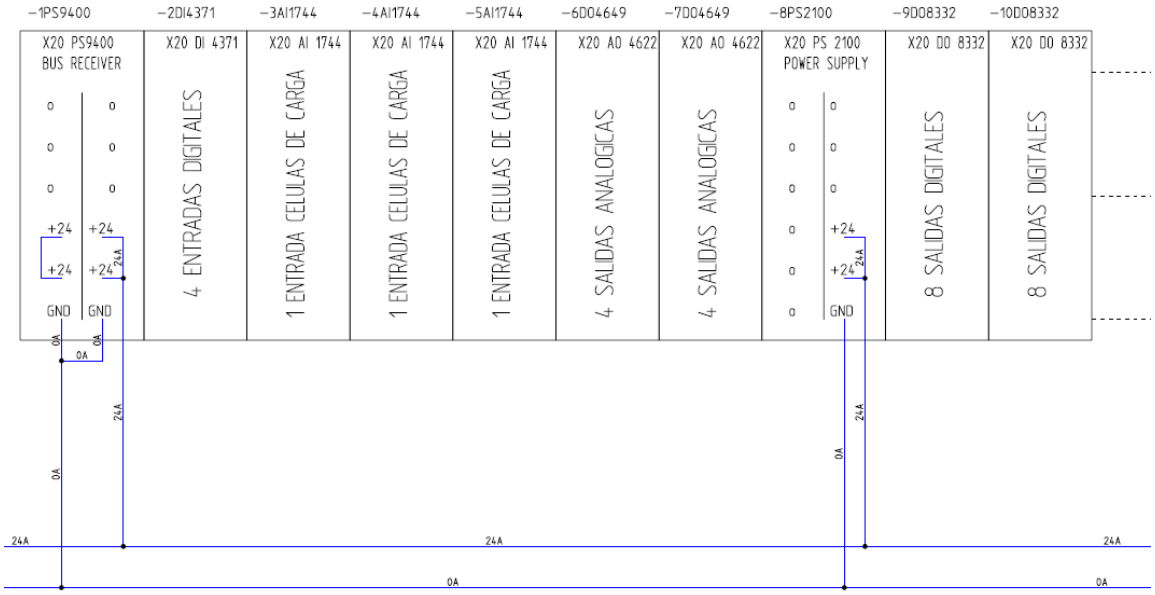


Ilustración 45

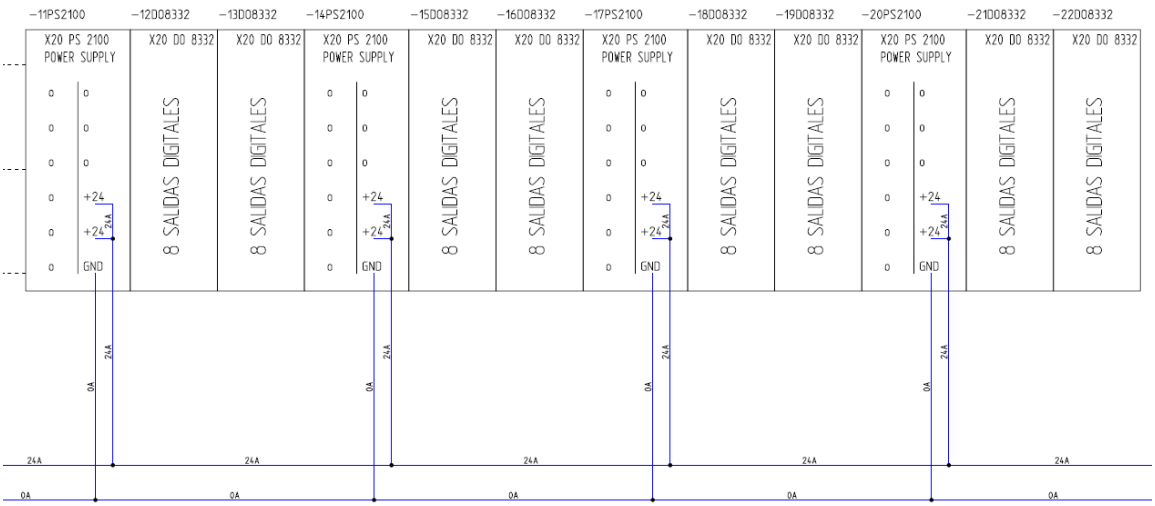


Ilustración 46

4.2.2. Distribución del hardware

Así pues, la distribución del hardware mencionado en el presente apartado, quedaría como la siguiente ilustración (ver también esquemas eléctricos del Anexo 1).

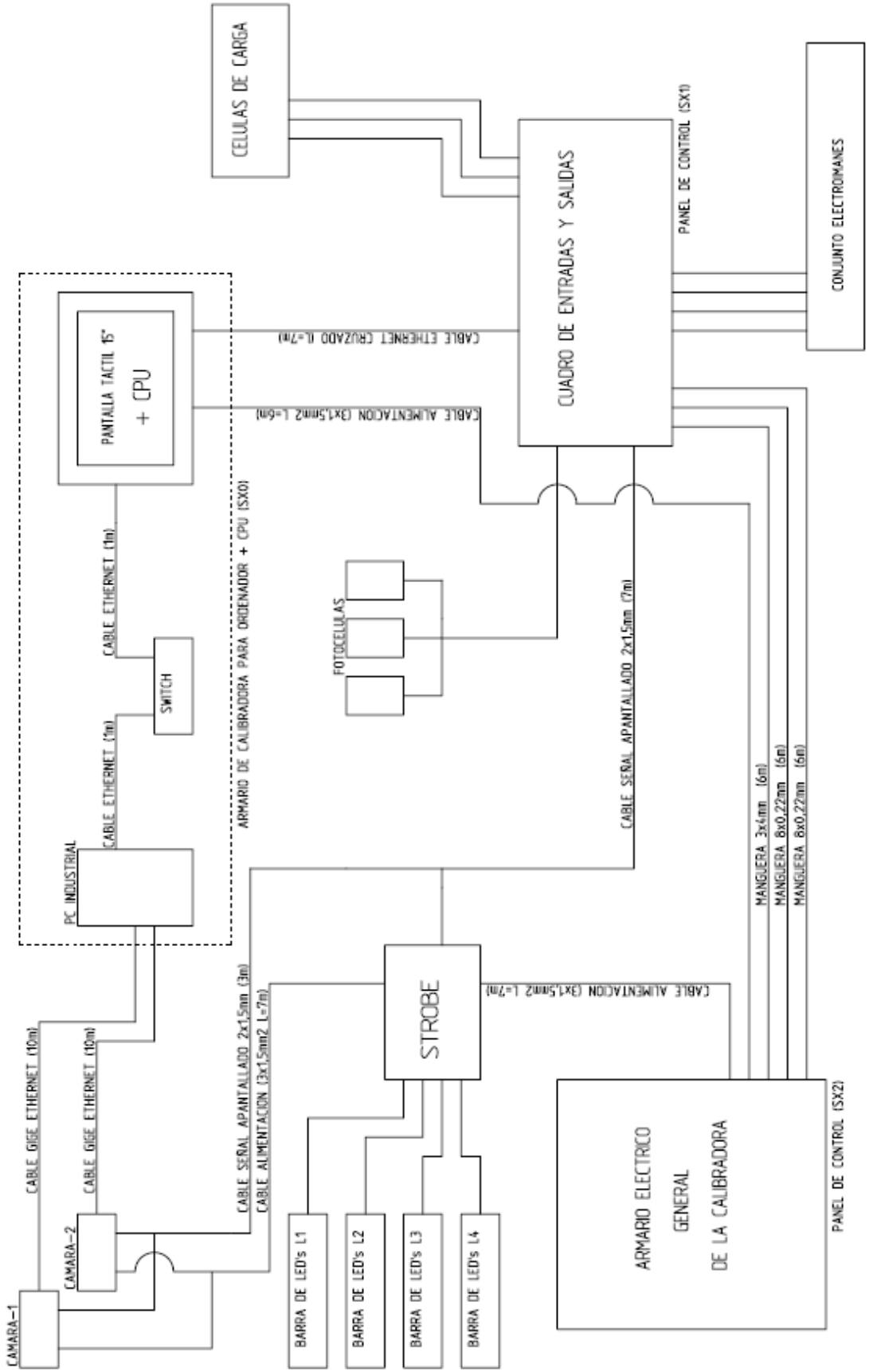
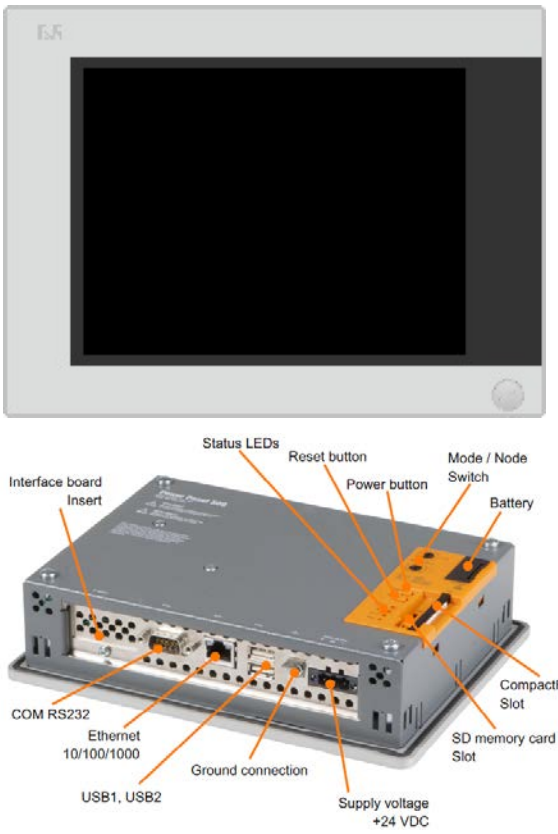


Ilustración 47

4.2.3. Procesador Industrial

Tal y cómo hemos mencionado utilizaremos la electrónica de Bernecker and Rainer B&R que nos ofrece las características que necesitamos en nuestra aplicación. En el caso del procesador industrial he elegido el Power Panel PP420 (4PP420.1043-75) que además de CPU también dispone de una pantalla táctil de 10 pulgadas TFT cuya función es la de permitir que el operario interactúe con la máquina.



Características técnicas:

- Número de serie: 4PP420.1043-75
- Power Panel 420 10" XGA TFT display con pantalla táctil (resistiva)
- Resolución: XGA, 1024 x 768 pixels
- Conexiones para 1x RS232, 3x USB 2.0, 1x Ethernet 10/100/1000
- Protección IP65 (en la parte frontal)
- Alimentación: 24 VDC ±25%, 1A (31W)
- Memória RAM: DDR2 SDRAM 2GB
- Memória de programa: tarjeta SD

Ilustración 48

Para el montaje de la pantalla hemos diseñado una caja mecánica guiándome por las medidas de la pantalla y la inclinación necesaria facilitada por el fabricante para la correcta visualización de esta

(Ver siguientes ilustraciones).

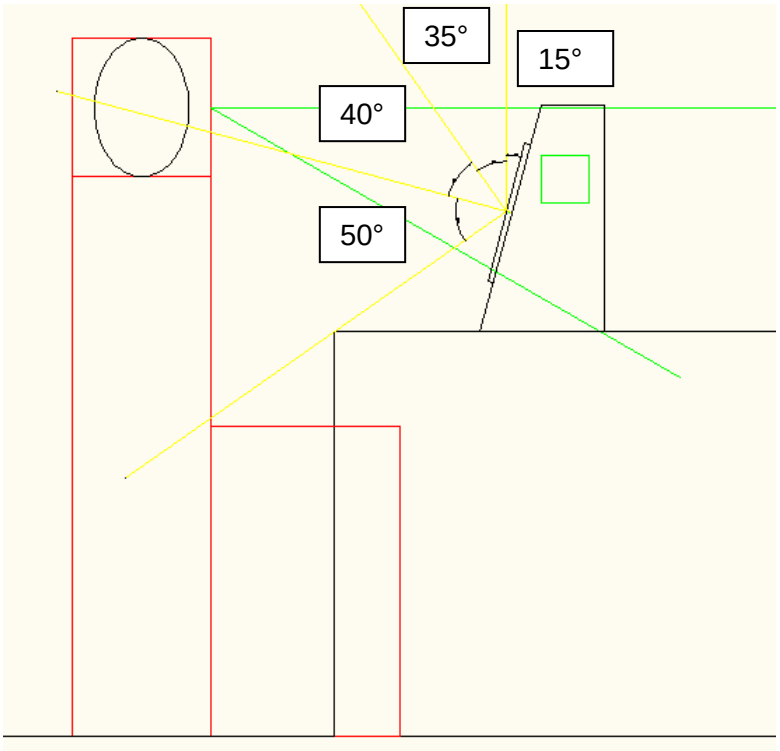


Ilustración 49

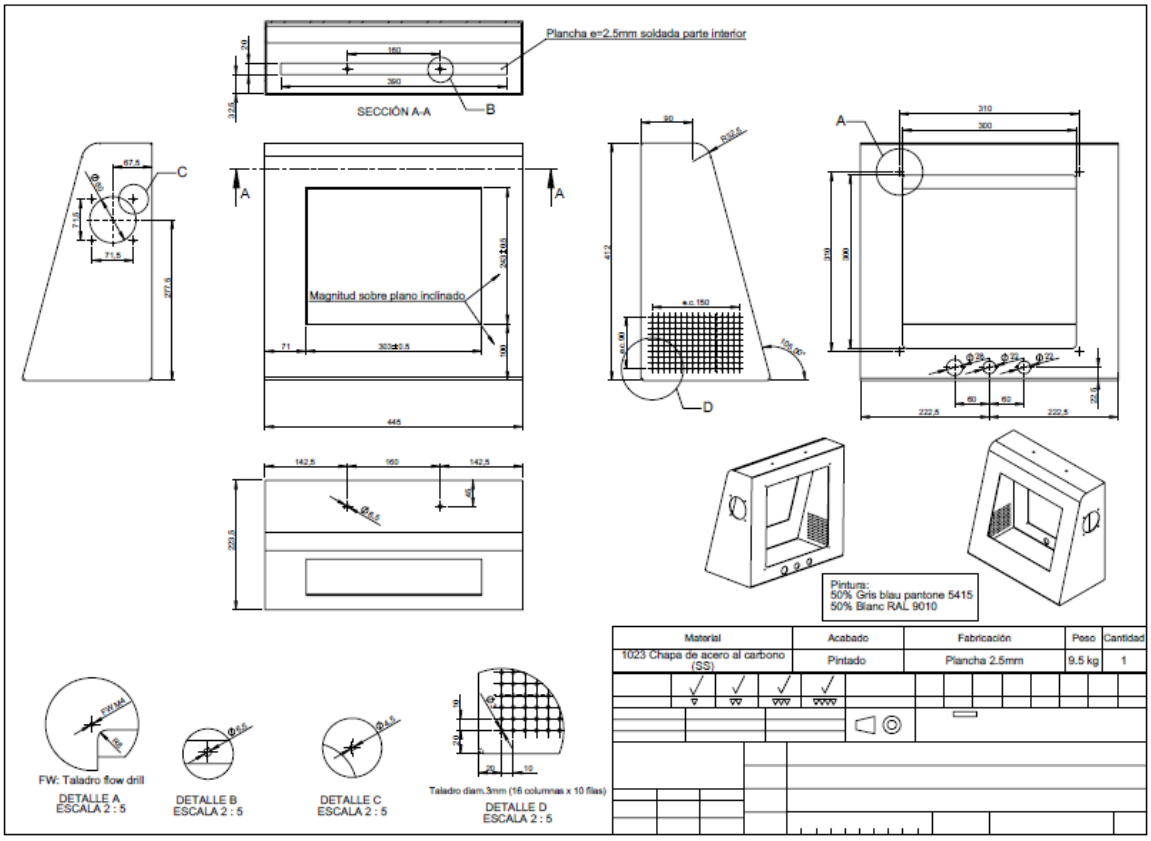


Ilustración 50

4.3. Software

4.3.1. Configuración del Hardware

Tal y como hemos expuesto en el apartado "4.2. Hardware" la electrónica de B&R dispone de una pantalla táctil con CPU incorporada que se comunica por X2X a través de una tarjeta de comunicación "3IF789.9-1" a una cabecera "X20PS2100" que a su vez se encuentra conectada a todos los módulos de entradas y salidas del PLC (ver siguiente ilustración).

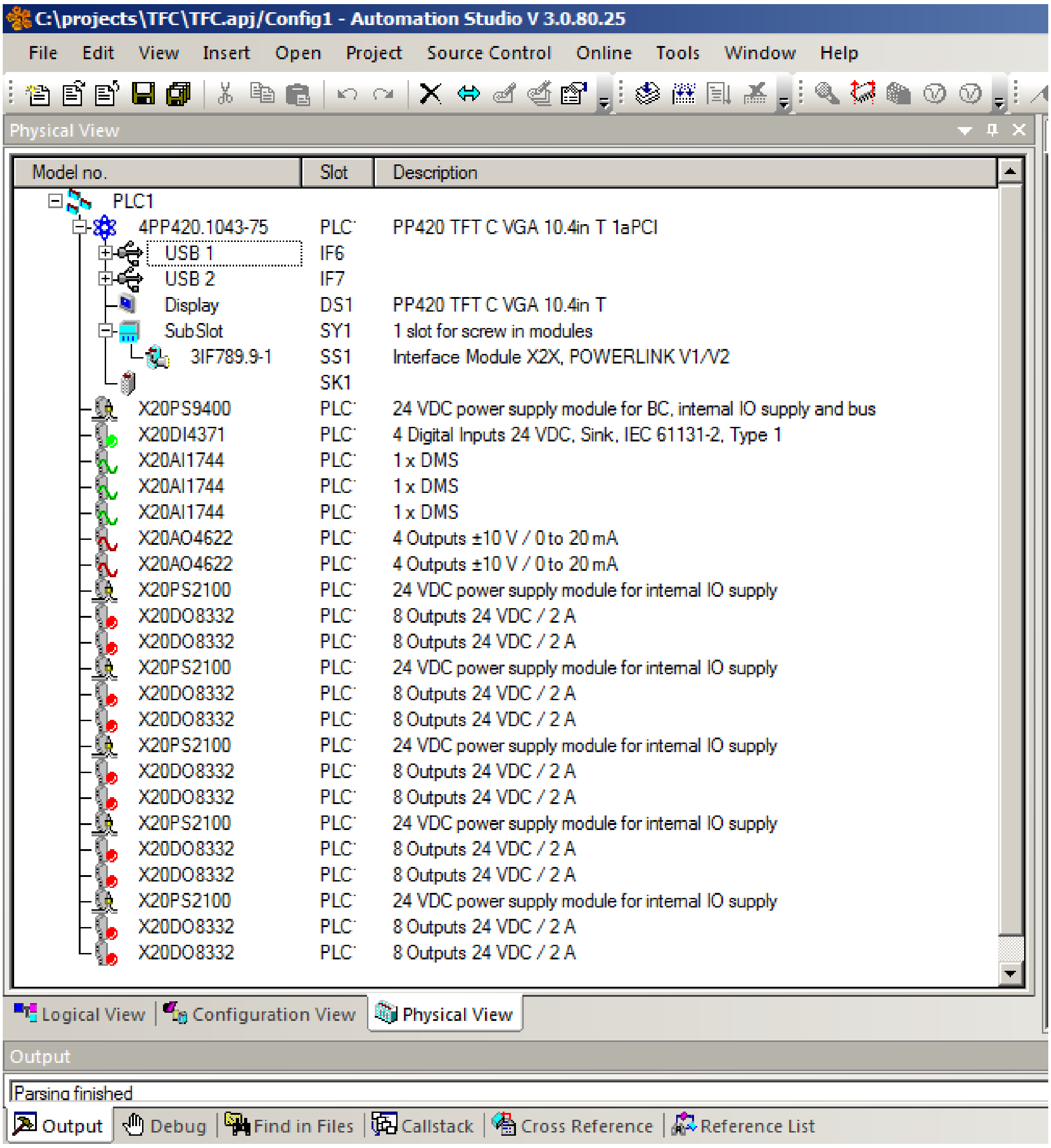


Ilustración 51

4.3.2. Módulos de programación

El programa dispone de los siguientes bloques de programación que vamos a detallar a continuación:

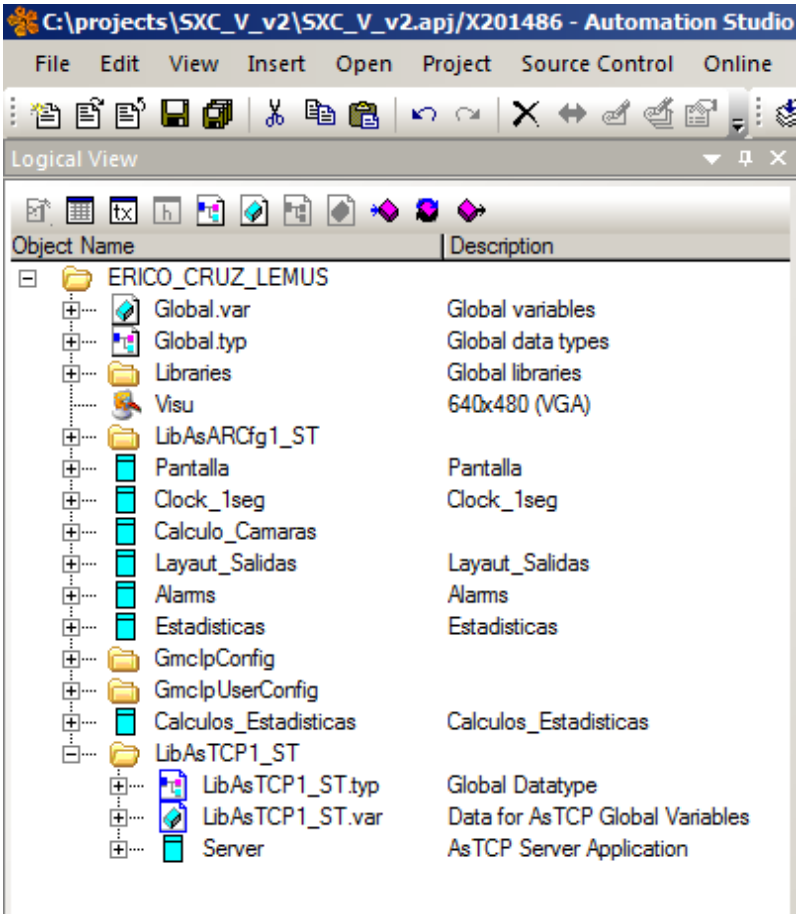


Ilustración 52

4.3.3. Bloque Pantalla

El módulo de pantalla es utilizado para programar la configuración de la pantalla táctil según los parámetros introducidos por el técnico que configura la calibradora durante la instalación (ver comentarios introducidos durante la programación resaltado en amarillo).

```
(*********************************************************************
 * COPYRIGHT --
 *********************************************************************
 * Program: Pantalla
 * File: PantallaCyclic.st
 * Author: Erico Cruz
 * Created: February 23, 2010
 *********************************************************************
 * Implementation of program Pantalla
 *********************************************************************)
```

```
PROGRAM _CYCLIC
(*---------------------------------------- Escoger pantalla para configurar electroimanes ----------------------------------------*)
 IF (Pantalla_Actual=200 OR Pantalla_Actual=202 OR Pantalla_Actual=203 OR Pantalla_Actual=204 OR
Pantalla_Actual=205 OR Pantalla_Actual=206 OR Pantalla_Actual=207 OR Pantalla_Actual=208) THEN
            Cambiar_Pantalla:=Pantalla_Config_Electroimanes;
            END_IF
 IF Config_Numero_Lineas=1 THEN
            Pantalla_Select_Conf_Electroiman[1]:=0;
            ELSE
            Pantalla_Select_Conf_Electroiman[1]:=1;
            END_IF
 IF Config_Numero_Lineas=2 THEN
            Pantalla_Select_Conf_Electroiman[2]:=0;
            ELSE
            Pantalla_Select_Conf_Electroiman[2]:=1;
            END_IF
 IF Config_Numero_Lineas=3 THEN
            Pantalla_Select_Conf_Electroiman[3]:=0;
            ELSE
            Pantalla_Select_Conf_Electroiman[3]:=1;
            END_IF
 IF Config_Numero_Lineas=4 THEN
            Pantalla_Select_Conf_Electroiman[4]:=0;
            ELSE
            Pantalla_Select_Conf_Electroiman[4]:=1;
            END_IF
 IF Config_Numero_Lineas=5 THEN
            Pantalla_Select_Conf_Electroiman[5]:=0;
            ELSE
            Pantalla_Select_Conf_Electroiman[5]:=1;
            END_IF
 IF Config_Numero_Lineas=6 THEN
            Pantalla_Select_Conf_Electroiman[6]:=0;
            ELSE
            Pantalla_Select_Conf_Electroiman[6]:=1;
            END_IF
 IF Config_Numero_Lineas=7 THEN
            Pantalla_Select_Conf_Electroiman[7]:=0;
            ELSE
            Pantalla_Select_Conf_Electroiman[7]:=1;
            END_IF
 IF Config_Numero_Lineas=8 THEN
            Pantalla_Select_Conf_Electroiman[8]:=0;
            ELSE
            Pantalla_Select_Conf_Electroiman[8]:=1;
            END_IF
```

```
(*---------------------------------- Pantallas emergentes avisos calibración básculas ----------------------------------*)
 IF EDGEPOS (Config_Reset_Basculas) THEN
            Pantalla_Emergente_basc_Reset:=0;
            ELSIF EDGEPOS (gControl.User.Reset) THEN
            Pantalla_Emergente_basc_Reset:=1;
            END_IF

 IF EDGEPOS(Config_Cero_Basculas) THEN
            Pantalla_Emergente_vel_trabajo:=0;
            END_IF
 (*---------------------------------------- Guardar y leer memoria de las diferentes variedades ------------------------*)
  IF EDGEPOS(Button_Variedad_UP) THEN
              Numero_Variedad:=Numero_Variedad+1;
            ELSIF EDGEPOS(Button_Variedad_DOWN) THEN
              Numero_Variedad:=Numero_Variedad-1;
            END_IF

 //Leer la variedad de la memoria
 IF (Button_Leer_Variedad=TRUE OR EDGEPOS(Button_Variedad_UP) OR EDGEPOS(Button_Variedad_DOWN))
THEN
            Variedad_de_la_Fruta_Lectura:=Variedad_de_la_Fruta_Memoria[Numero_Variedad];
   END_IF
  //Guardar la variedad en la memoria
 IF Button_Guardar_Variedad=TRUE THEN
            Variedad_de_la_Fruta_Memoria[Numero_Variedad]:=Variedad_de_la_Fruta_Lectura;
            Pantalla_Emerg_Guardar_Variedad:=1;
   END_IF
 (*---------------------------------------- Guardar y leer memoria de los diferentes grados ----------------------------------*)
 IF (EDGEPOS(Button_Grado_UP)) THEN
              Numero_Grado:=Numero_Grado+1;
            ELSIF (EDGEPOS(Button_Grado_DOWN)) THEN
              Numero_Grado:=Numero_Grado-1;
   END_IF

 IF (Button_Leer_Grado=TRUE OR EDGEPOS(Button_Grado_UP) OR EDGEPOS(Button_Grado_DOWN)) THEN
            Grado_de_la_Fruta_Lectura:=Grado_de_la_Fruta_Memoria[Numero_Grado];
   END_IF

 IF Button_Guardar_Grado=TRUE THEN
            Grado_de_la_Fruta_Memoria[Numero_Grado]:=Grado_de_la_Fruta_Lectura;
            Pantalla_Emerg_Guardar_Grado:=1;
   END_IF
(*------------------------------------------- CONTROL CONTRASEÑAS ----------------------------------------------------------*)
// Button_Mantenimiento, Button_Cancelar_Password y Pantalla_Emergente_Password se utilizan en las pantallas del
operario
 IF EDGEPOS(Button_Mantenimiento) AND Nivel_Password_Actual>=1 THEN
            Cambiar_Pantalla:=300;
            ELSIF EDGEPOS(Button_Mantenimiento) AND Nivel_Password_Actual=0 THEN
            Pantalla_Emergente_Password:=0;
```

```
            ELSIF EDGEPOS(Button_Mantenimiento_Avanzado) THEN
            Pantalla_Emergente_Password_2:=0;
            END_IF
// Button_Mantenimiento_Avanzado, Button_Cancelar_Password_2 y Pantalla_Emergente_Password_2 se utilizan en las pantallas de configuración
 IF EDGEPOS(Button_Cancelar_Password) THEN
            Nivel_Password_Actual:=0;
            Pantalla_Emergente_Password:=1;
            ELSIF EDGEPOS(Button_Cancelar_Password_2) THEN
            Nivel_Password_Actual:=0;
            Pantalla_Emergente_Password_2:=1;
            END_IF
// Temporizador para poner el nivel del Password a cero
 IF Nivel_Password_Actual > 0 THEN
            Password_Activado:=TRUE;
            END_IF
 IF EDGEPOS(Timer_Password.Q) THEN
            Nivel_Password_Actual:=0;
            Password_Activado:=FALSE;
   END_IF
 IF Config_Tiempo_Password < 20000 THEN
            Config_Tiempo_Password := 20000;
            END_IF
  Timer_Password(IN := Password_Activado, PT := UDINT_TO_TIME(Config_Tiempo_Password));

 (*---------------------------------------- MEMORIZAR LOS DATOS DEL AGRICULTOR ----------------------------------------------*)
 IF EDGEPOS(Button_Agricultor_UP) THEN
            Numero_Agricultor:=Numero_Agricultor+1;
            Leer_Agricultor:=TRUE;
            ELSIF EDGEPOS(Button_Agricultor_DOWN) THEN
            Numero_Agricultor:=Numero_Agricultor-1;
            Leer_Agricultor:=TRUE;
            ELSIF EDGEPOS(Button_Leer_Agricultor) THEN
            Leer_Agricultor:=TRUE;
 END_IF

 IF EDGEPOS(Button_Guardar_Agricultor) THEN
            Pantalla_Emerg_Guarda_Agricultor :=1;
            Guardar_Agricultor:=TRUE;
 END_IF
 (*---------------------------------------- GUARDAR LOS DATOS DEL AGRICULTOR ----------------------------------------------*)
 IF Guardar_Agricultor=TRUE THEN

            Datos_Agricultor[Numero_Agricultor].Nombre_del_Agricultor := Estadisticas_por_Calibre.Nombre_Agricultor;
            Datos_Agricultor[Numero_Agricultor].Finca              := Estadisticas_por_Calibre.Nombre_Finca;
            Datos_Agricultor[Numero_Agricultor].Variedad_de_la_Fruta  := Estadisticas_por_Calibre.Nombre_Variedad;
            Datos_Agricultor[Numero_Agricultor].Modo_de_Calibracion   :=
Estadisticas_por_Calibre.Modo_de_Calibracion;
```

```
            Pantalla_Emerg_Guarda_Agricultor:=1;
            Guardar_Agricultor:=FALSE;
END_IF
(*------------------------------------ LEER LOS DATOS DEL AGRICULTOR ----------------------------------------------------*)
IF Leer_Agricultor=TRUE THEN

            Estadisticas_por_Calibre.Nombre_Agricultor   := Datos_Agricultor[Numero_Agricultor].Nombre_del_Agricultor;
            Estadisticas_por_Calibre.Nombre_Finca        := Datos_Agricultor[Numero_Agricultor].Finca;
            Estadisticas_por_Calibre.Nombre_Variedad     := Datos_Agricultor[Numero_Agricultor].Variedad_de_la_Fruta;
            Estadisticas_por_Calibre.Modo_de_Calibracion := Datos_Agricultor[Numero_Agricultor].Modo_de_Calibracion;
            Leer_Agricultor:=FALSE;
END_IF
END_PROGRAM
```

4.3.4. Clock 1 segundo

Con este módulo generamos pulsaciones en un bit con un periodo de 1 segundo para utilizarlo en numerosas ocasiones en la implementación de todo el programa. Para ello requeriremos programar este módulo para que se ejecute cada 1000ms (1 segundo).

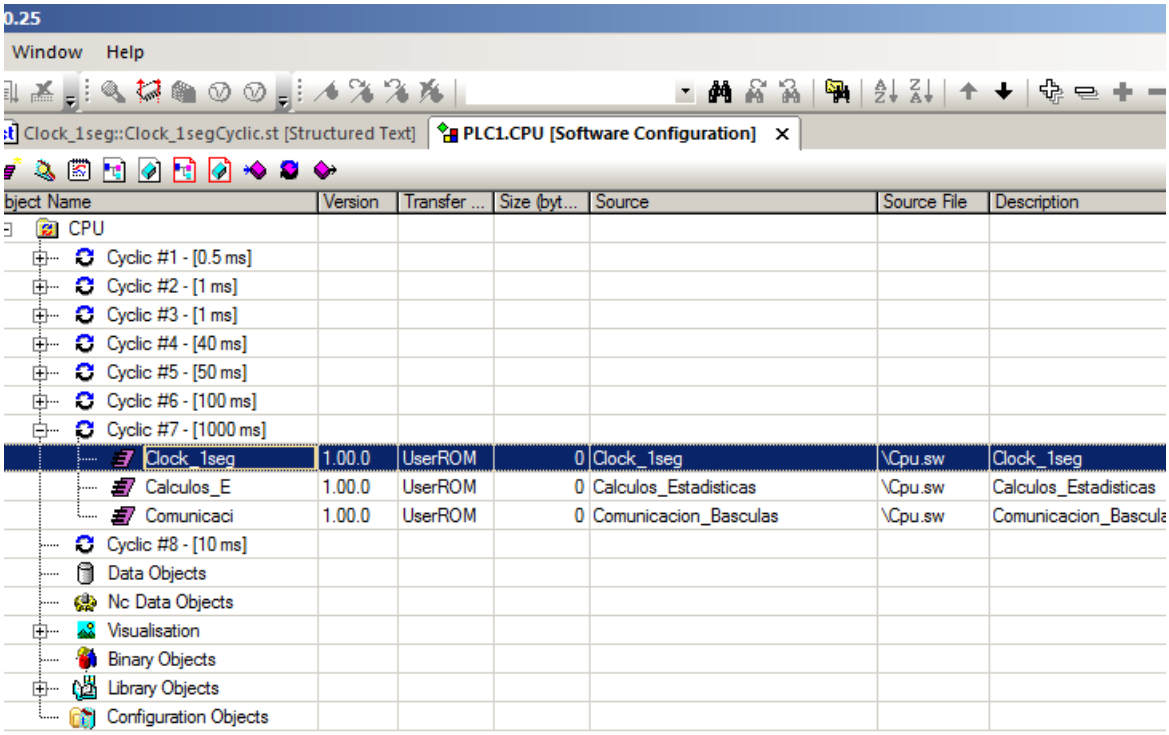


Ilustración 53

```
(*********************************************************************
 * COPYRIGHT --
 *********************************************************************
 * Program: Clock_1seg
 * File: Clock_1segCyclic.st
 * Author: Erico Cruz
 * Created: February 23, 2010
 *********************************************************************
 * Implementation of program Clock_1seg
 *********************************************************************)
(*------------------------- GENERAMOS LOS PULSOS CADA VEZ QUE SE EJECUTA EL PROGRAMA----------------------*)
PROGRAM _CYCLIC
Clock_1seg := NOT Clock_1seg;
END_PROGRAM
```

4.3.5. Cálculo cámaras

Este es uno de los módulos más importantes que requerimos para nuestra aplicación debido a que este módulo, con los datos recibidos de la CPU de las cámaras, cada vez que reciba los pulsos de detección de paso de cazoleta (a través de las fotocélulas) realizará los cálculos necesarios para asignarle a la fruta la salida según lo configurado por el operario en la pantalla táctil.

```
(*********************************************************************
 * COPYRIGHT -- Bernecker + Rainer
 *********************************************************************
 * Program: Caluculo_Camaras
 * File: Caluculo_Camaras.st
 * Author: Erico Cruz
 * Created: March 12, 2010
 *********************************************************************
 * Implementation of program Caluculo_Camaras
 *********************************************************************)
PROGRAM _INIT
(*---------------------------------------- INICIALIZAMOS LAS VARIABLES ----------------------------------------*)
  FOR j:=1 TO (Config_Numero_Lineas) DO
                contador_posicion_1[j]:=1;
                contador_posicion_2[j]:=1;
                contador_posicion_3[j]:=1;
                contador_posicion_4[j]:=1;
  END_FOR
END_PROGRAM

PROGRAM _CYCLIC
(*--------- EJECUTAMOS CUANDO RECIBIMOS SEÑAL DE DETECCICÓN DE PASO DE CAZOLETA ----------------------*)
IF EDGEPOS(gControl.User.WeighPerformed) THEN
```

```
(* ---------------------------------- CÁLCULO CÁMARAS LINEA 1-4 ---------------------------------------------------------------------- *)
 IF (Pantalla_Layaut.Modo_de_Calibracion=3 OR Pantalla_Layaut.Modo_de_Calibracion=6) THEN // Calibramos con diámetro
            FOR j:=1 TO (Config_Numero_Lineas) DO //Para el número de línea
(* Calibración por diámetro Mayor *)
                    IF Select_Modo_Diametro=0 THEN // calibración por diámetro mayor

        IF Camaras_Grado.Linea[j].Posicion_4.Diametro_Max > Average.Linea[j].Posicion_3.Diametro THEN Average.Linea[j].Posicion_4.Diametro:=Camaras_Grado.Linea[j].Posicion_4.Diametro_Max+ Desviacion_Diametro[j].Diametro_Maximo;
        ELSIF Camaras_Grado.Linea[j].Posicion_4.Diametro_Max <= Average.Linea[j].Posicion_3.Diametro THEN
                                Average.Linea[j].Posicion_4.Diametro:=Average.Linea[j].Posicion_3.Diametro + Desviacion_Diametro[j].Diametro_Maximo;
                                END_IF
         IF Camaras_Grado.Linea[j].Posicion_3.Diametro_Max > Average.Linea[j].Posicion_2.Diametro THEN
                                Average.Linea[j].Posicion_3.Diametro := Camaras_Grado.Linea[j].Posicion_3.Diametro_Max;
        ELSIF Camaras_Grado.Linea[j].Posicion_3.Diametro_Max <= Average.Linea[j].Posicion_2.Diametro THEN
                                Average.Linea[j].Posicion_3.Diametro:=Average.Linea[j].Posicion_2.Diametro;
                                END_IF
        IF Camaras_Grado.Linea[j].Posicion_2.Diametro_Max > Average.Linea[j].Posicion_1.Diametro THEN
                                Average.Linea[j].Posicion_2.Diametro := Camaras_Grado.Linea[j].Posicion_2.Diametro_Max;
        ELSIF Camaras_Grado.Linea[j].Posicion_2.Diametro_Max <= Average.Linea[j].Posicion_1.Diametro THEN
                                Average.Linea[j].Posicion_2.Diametro:=Average.Linea[j].Posicion_1.Diametro;
                                END_IF
        Average.Linea[j].Posicion_1.Diametro := Camaras_Grado.Linea[j].Posicion_1.Diametro_Max;
                    END_IF

(* Calibración por diámetro Menor *)
                IF Select_Modo_Diametro=1 THEN // calibración por diámetro menor

        IF (((Camaras_Grado.Linea[j].Posicion_4.Diametro_Min < Average.Linea[j].Posicion_3.Diametro) OR (Average.Linea[j].Posicion_3.Diametro = 0)) AND (Camaras_Grado.Linea[j].Posicion_4.Diametro_Min >= DiametroMinimo)) THEN
        Average.Linea[j].Posicion_4.Diametro := Camaras_Grado.Linea[j].Posicion_4.Diametro_Min + Desviacion_Diametro[j].Diametro_Minimo;
        ELSIF Camaras_Grado.Linea[j].Posicion_4.Diametro_Min >= Average.Linea[j].Posicion_3.Diametro THEN
        Average.Linea[j].Posicion_4.Diametro :=Average.Linea[j].Posicion_3.Diametro + Desviacion_Diametro[j].Diametro_Minimo;
                                END_IF

         IF (((Camaras_Grado.Linea[j].Posicion_3.Diametro_Min < Average.Linea[j].Posicion_2.Diametro) OR (Average.Linea[j].Posicion_2.Diametro = 0)) AND (Camaras_Grado.Linea[j].Posicion_3.Diametro_Min >= DiametroMinimo)) THEN
        Average.Linea[j].Posicion_3.Diametro := Camaras_Grado.Linea[j].Posicion_3.Diametro_Min;
        ELSIF Camaras_Grado.Linea[j].Posicion_3.Diametro_Min >= Camaras_Grado.Linea[j].Posicion_2.Diametro THEN
        Average.Linea[j].Posicion_3.Diametro := Average.Linea[j].Posicion_2.Diametro;
```

```
                    END_IF
        IF (((Camaras_Grado.Linea[j].Posicion_2.Diametro_Min < Average.Linea[j].Posicion_1.Diametro) OR
(Average.Linea[j].Posicion_1.Diametro = 0)) AND (Camaras_Grado.Linea[j].Posicion_2.Diametro_Min >=
DiametroMinimo)) THEN
        Average.Linea[j].Posicion_2.Diametro := Camaras_Grado.Linea[j].Posicion_2.Diametro_Min;
        ELSIF Camaras_Grado.Linea[j].Posicion_2.Diametro_Min >= Average.Linea[j].Posicion_1.Diametro THEN
                    Average.Linea[j].Posicion_2.Diametro := Average.Linea[j].Posicion_1.Diametro;
                    END_IF
        IF (Camaras_Grado.Linea[j].Posicion_1.Diametro_Min >= DiametroMinimo) THEN
        Average.Linea[j].Posicion_1.Diametro := Camaras_Grado.Linea[j].Posicion_1.Diametro_Min;
                    ELSE
                    Average.Linea[j].Posicion_1.Diametro := 0;
                    END_IF
                END_IF

(* Calibración por diámetro Medio*)
                IF Select_Modo_Diametro=2 THEN // calibración por diámetro Medio
            (* POSICION CAZOLETA 4 *)
                    IF (Camaras_Grado.Linea[j].Posicion_4.Diametro_Avg >= DiametroMinimo) THEN
                            contador_posicion_4[j] := contador_posicion_3[j] + 1;
                      Average.Linea[j].Posicion_4.Diametro :=
(Camaras_Grado.Linea[j].Posicion_4.Diametro_Avg + Average.Linea[j].Posicion_3.Diametro)/(contador_posicion_4[j]) +
Desviacion_Diametro[j].Diametro_Medio;
            ELSE
                            contador_posicion_4[j] := contador_posicion_3[j];
                            Average.Linea[j].Posicion_4.Diametro :=
(Average.Linea[j].Posicion_3.Diametro)/(contador_posicion_4[j]) + Desviacion_Diametro[j].Diametro_Medio;
                    END_IF
            (* POSICION CAZOLETA 3 *)
                    IF (Camaras_Grado.Linea[j].Posicion_3.Diametro_Avg >= DiametroMinimo) THEN
                      Average.Linea[j].Posicion_3.Diametro :=
(Camaras_Grado.Linea[j].Posicion_3.Diametro_Avg + Average.Linea[j].Posicion_2.Diametro);
                      contador_posicion_3[j] := contador_posicion_2[j] + 1;
                    ELSE
                            Average.Linea[j].Posicion_3.Diametro := Average.Linea[j].Posicion_2.Diametro;
                            contador_posicion_3[j] := contador_posicion_2[j];
                    END_IF
            (* POSICION CAZOLETA 2 *)
                    IF (Camaras_Grado.Linea[j].Posicion_2.Diametro_Avg >= DiametroMinimo) THEN
                            Average.Linea[j].Posicion_2.Diametro :=
(Camaras_Grado.Linea[j].Posicion_2.Diametro_Avg + Average.Linea[j].Posicion_1.Diametro);
                      contador_posicion_2[j] := contador_posicion_1[j] + 1;
                    ELSE
                            Average.Linea[j].Posicion_2.Diametro := Average.Linea[j].Posicion_1.Diametro;
                            contador_posicion_2[j] := contador_posicion_1[j];
                    END_IF
```

```
                    (* POSICION CAZOLETA 1 *)
                            IF (Camaras_Grado.Linea[j].Posicion_1.Diametro_Avg >= DiametroMinimo) THEN
                                    Average.Linea[j].Posicion_1.Diametro :=
(Camaras_Grado.Linea[j].Posicion_1.Diametro_Avg);
                                    contador_posicion_1[j] := 1;
                            ELSE
                                    Average.Linea[j].Posicion_1.Diametro := 1;
                            END_IF
        END_IF
            END_FOR
                    Hacer_Calculos:=TRUE;
(*****************************************************************************************************************)
(* Guardamos el diámetro esperando el número de posiciones para llegar a las básculas y calcular la salida *)
(*****************************************************************************************************************)
                    PositionOffset := UINT_TO_REAL((gLine[p].ActPosition-1)) +
INT_TO_REAL(gControlExit[p].User.PositionExit[0].NumberPulses); (* PositionOffset = (ActPos-1) + PosExit(0) *)

      IF PositionOffset >= gLine[p].User.NumPositions THEN  (* (ActPos-1) + PosExit(0) > MaxPos *)
        Offset := ( INT_TO_REAL(gControlExit[p].User.PositionExit[0].NumberPulses) +
UINT_TO_REAL(gLine[p].ActPosition-1) ) - UINT_TO_REAL(gLine[p].User.NumPositions)  ;  (* Offset = (PosExit(i) +
(ActPos-1)) - MaxPos *)
      ELSE
        Offset := UINT_TO_REAL(gLine[p].ActPosition-1) +
INT_TO_REAL(gControlExit[p].User.PositionExit[0].NumberPulses); (* Offset = (ActPos-1) + PosExit(i) *)
      END_IF
             IF Average.Linea[p].Posicion_4.Diametro >= DiametroMinimo THEN
                    gControlExit[p].User.Grado[REAL_TO_UINT(Offset)]    :=
INT_TO_USINT(Average.Linea[p].Posicion_4.Diametro); // Para visualizar en la tarea Visu1-Visu8 los diámetros
                    gControlExit[p].User.Diametro[REAL_TO_UINT(Offset)] :=  Average.Linea[p].Posicion_4.Diametro;
                  ELSIF Average.Linea[p].Posicion_4.Diametro < DiametroMinimo THEN
               gControlExit[p].User.Grado[REAL_TO_UINT(Offset)]    := 0; // Para visualizar en la tarea Visu1-Visu8 los
diámetros
                    gControlExit[p].User.Diametro[REAL_TO_UINT(Offset)] := 0;
                  END_IF
          (*****************************************************************************************************************)
   END_FOR
 END_IF
 Trigger_Camaras:=FALSE;
 END_IF
(************************************** calculos calibres y salidas ***************************************************)
  FOR p:=1 TO (Config_Numero_Lineas) DO
          Calculos_Linea[p].Peso_Actual:= REAL_TO_INT((gLine[p].WeighPosition[gLine[p].ActPosition-1])*1000);
          Diametro[p]:= gControlExit[p].User.Diametro[gLine[p].ActPosition-1];
          Calculos_Linea[p].Salida_Asignada:=FALSE
          IF Hacer_Test_Completo=TRUE THEN
                    FOR i:=1 TO 4 DO
               Calculos_Linea[p].Numero_de_Salida:=0;
                    END_FOR
          END_IF
```

```
    END_FOR
(*------------------------------------------- CÁLCULO DEL CALIBRE ------------------------------------------------------*)
IF Hacer_Test_Completo=FALSE AND Button_Test_por_Salida=FALSE THEN
// Cálculo del calibre por DIÁMETRO
 IF (Pantalla_Layaut.Modo_de_Calibracion=3 OR Pantalla_Layaut.Modo_de_Calibracion=4) THEN
          FOR p:=1 TO (Config_Numero_Lineas) DO //Para número de línea
                    FOR m:=1 TO (INT_TO_USINT(Variedad_de_la_Fruta_Lectura.Numero_de_Grados) + 1) DO //Para
número del calibre
                              IF m < (Variedad_de_la_Fruta_Lectura.Numero_de_Grados + 1) THEN
                                IF (Diametro[p] >= Variedad_de_la_Fruta_Lectura.Diametro_del_Grado[m]) AND
(Diametro[p] < Variedad_de_la_Fruta_Lectura.Diametro_del_Grado[m+1] AND Calculos_Linea[p].Peso_Actual >
(1000*Config_TarasOffset)) THEN
                                Calculos_Linea[p].Numero_de_Calibre:=m;
                                          // sumamos los valores para las estadísticas

Estadisticas_por_Calibre.Peso_por_Calibre_Gramos[m]:=Estadisticas_por_Calibre.Peso_por_Calibre_Gramos[m]+Calcul
os_Linea[p].Peso_Actual;
Estadisticas_por_Calibre.Piezas_por_Calibre[m]:=Estadisticas_por_Calibre.Piezas_por_Calibre[m]+1;
                                  END_IF
                                END_IF
                                IF m = (Variedad_de_la_Fruta_Lectura.Numero_de_Grados) AND (Diametro[p] >=
Variedad_de_la_Fruta_Lectura.Diametro_del_Grado[m]) AND Calculos_Linea[p].Peso_Actual > 15 THEN
                                  Calculos_Linea[p].Numero_de_Calibre:=m;
                                          // sumamos los valores para las estadísticas

Estadisticas_por_Calibre.Peso_por_Calibre_Gramos[m]:=Estadisticas_por_Calibre.Peso_por_Calibre_Gramos[m]+Calcul
os_Linea[p].Peso_Actual;
Estadisticas_por_Calibre.Piezas_por_Calibre[m]:=Estadisticas_por_Calibre.Piezas_por_Calibre[m]+1;
                                  END_IF
                              END_FOR
          END_FOR
 END_IF
(*------------------------------------------------------- ENVIAR LOS DATOS DE LAS SALIDAS ---------------------------------*)
    FOR p:=1 TO (Config_Numero_Lineas) DO
          gControlExit[p].User.ExitForcedPosition_Auto := Calculos_Linea[p].Numero_de_Salida;
          IF Calculos_Linea[p].Salida_Asignada = TRUE THEN
            gVisu[p].LastExit[0] := USINT_TO_REAL(Calculos_Linea[p].Numero_de_Salida);
          END_IF
          gLine[p].User.GetData:=TRUE;
          END_FOR
 gControl.User.WeighPerformed:=FALSE;
          FOR p:=1 TO 4 DO
            Calculos_Linea[p].Numero_de_Salida:=0;
                    END_FOR
          FOR p:=1 TO 16 DO
            Calculos_Linea[p].Numero_de_Calibre:=0;
                    END_FOR
 END_IF
END_PROGRAM
```

4.3.6. Layaut salidas

Este modulo será el que contenga el código de programa para la gestión de la pantalla (que interactúa con el operario de la calibradora) de asignación de las salidas de cada calibre.

```
(*********************************************************************
 * COPYRIGHT --
 *********************************************************************
 * Program: Layaut_Salidas
 * File: Layaut_Salidas.st
 * Author: Erico Cruz
 * Created: March 17, 2010
 *********************************************************************
 * Implementation of program Layaut_Salidas
 *********************************************************************)

PROGRAM _INIT
(*---------------------------------- INICIALIZAMOS VARIABLES ----------------------*)
Grado[1]:='A';
Grado[2]:='B';
Grado[3]:='C';
Grado[4]:='D';
Grado[5]:='E';
Grado[6]:='T';

END_PROGRAM

PROGRAM _CYCLIC
(*---- SI EL OPERARIO PULSA UN BOTON DE ASIGNACIÓN DE SALIDA SE RESETEAN LOS OTROS BOTONES ---*)
 IF EDGEPOS(Button_Select_Salida[1]) THEN
            Button_Select_Salida[2]:=FALSE;
            Button_Select_Salida[3]:=FALSE;
            Button_Select_Salida[4]:=FALSE;
            Button_Select_Salida[5]:=FALSE;
            Button_Select_Salida[6]:=FALSE;
            Button_Select_Salida[7]:=FALSE;
            Button_Select_Salida[8]:=FALSE;
            Button_Select_Salida[9]:=FALSE;
            Button_Select_Salida[10]:=FALSE;
            Button_Select_Salida[11]:=FALSE;
            Button_Select_Salida[12]:=FALSE;
            Button_Select_Salida[13]:=FALSE;
            Button_Select_Salida[14]:=FALSE;
            Button_Select_Salida[15]:=FALSE;
            Button_Select_Salida[16]:=FALSE;
            END_IF
```

```
IF EDGEPOS(Button_Select_Salida[2]) THEN
        Button_Select_Salida[1]:=FALSE;
        Button_Select_Salida[3]:=FALSE;
        Button_Select_Salida[4]:=FALSE;
        Button_Select_Salida[5]:=FALSE;
        Button_Select_Salida[6]:=FALSE;
        Button_Select_Salida[7]:=FALSE;
        Button_Select_Salida[8]:=FALSE;
        Button_Select_Salida[9]:=FALSE;
        Button_Select_Salida[10]:=FALSE;
        Button_Select_Salida[11]:=FALSE;
        Button_Select_Salida[12]:=FALSE;
        Button_Select_Salida[13]:=FALSE;
        Button_Select_Salida[14]:=FALSE;
        Button_Select_Salida[15]:=FALSE;
        Button_Select_Salida[16]:=FALSE;
        END_IF

IF EDGEPOS(Button_Select_Salida[3]) THEN
        Button_Select_Salida[1]:=FALSE;
        Button_Select_Salida[2]:=FALSE;
        Button_Select_Salida[4]:=FALSE;
        Button_Select_Salida[5]:=FALSE;
        Button_Select_Salida[6]:=FALSE;
        Button_Select_Salida[7]:=FALSE;
        Button_Select_Salida[8]:=FALSE;
        Button_Select_Salida[9]:=FALSE;
        Button_Select_Salida[10]:=FALSE;
        Button_Select_Salida[11]:=FALSE;
        Button_Select_Salida[12]:=FALSE;
        Button_Select_Salida[13]:=FALSE;
        Button_Select_Salida[14]:=FALSE;
        Button_Select_Salida[15]:=FALSE;
        Button_Select_Salida[16]:=FALSE;
        END_IF

IF EDGEPOS(Button_Select_Salida[4]) THEN
        Button_Select_Salida[1]:=FALSE;
        Button_Select_Salida[2]:=FALSE;
        Button_Select_Salida[3]:=FALSE;
        Button_Select_Salida[5]:=FALSE;
        Button_Select_Salida[6]:=FALSE;
        Button_Select_Salida[7]:=FALSE;
        Button_Select_Salida[8]:=FALSE;
        Button_Select_Salida[9]:=FALSE;
        Button_Select_Salida[10]:=FALSE;
        Button_Select_Salida[11]:=FALSE;
        Button_Select_Salida[12]:=FALSE;
        Button_Select_Salida[13]:=FALSE;
```

```
        Button_Select_Salida[14]:=FALSE;
        Button_Select_Salida[15]:=FALSE;
        Button_Select_Salida[16]:=FALSE;
        END_IF

IF EDGEPOS(Button_Select_Salida[5]) THEN
        Button_Select_Salida[1]:=FALSE;
        Button_Select_Salida[2]:=FALSE;
        Button_Select_Salida[3]:=FALSE;
        Button_Select_Salida[4]:=FALSE;
        Button_Select_Salida[6]:=FALSE;
        Button_Select_Salida[7]:=FALSE;
        Button_Select_Salida[8]:=FALSE;
        Button_Select_Salida[9]:=FALSE;
        Button_Select_Salida[10]:=FALSE;
        Button_Select_Salida[11]:=FALSE;
        Button_Select_Salida[12]:=FALSE;
        Button_Select_Salida[13]:=FALSE;
        Button_Select_Salida[14]:=FALSE;
        Button_Select_Salida[15]:=FALSE;
        Button_Select_Salida[16]:=FALSE;
        END_IF

IF EDGEPOS(Button_Select_Salida[6]) THEN
        Button_Select_Salida[1]:=FALSE;
        Button_Select_Salida[2]:=FALSE;
        Button_Select_Salida[3]:=FALSE;
        Button_Select_Salida[4]:=FALSE;
        Button_Select_Salida[5]:=FALSE;
        Button_Select_Salida[7]:=FALSE;
        Button_Select_Salida[8]:=FALSE;
        Button_Select_Salida[9]:=FALSE;
        Button_Select_Salida[10]:=FALSE;
        Button_Select_Salida[11]:=FALSE;
        Button_Select_Salida[12]:=FALSE;
        Button_Select_Salida[13]:=FALSE;
        Button_Select_Salida[14]:=FALSE;
        Button_Select_Salida[15]:=FALSE;
        Button_Select_Salida[16]:=FALSE;
        END_IF

IF EDGEPOS(Button_Select_Salida[7]) THEN
        Button_Select_Salida[1]:=FALSE;
        Button_Select_Salida[2]:=FALSE;
        Button_Select_Salida[3]:=FALSE;
        Button_Select_Salida[4]:=FALSE;
        Button_Select_Salida[5]:=FALSE;
        Button_Select_Salida[6]:=FALSE;
        Button_Select_Salida[8]:=FALSE;
```

```
        Button_Select_Salida[9]:=FALSE;
        Button_Select_Salida[10]:=FALSE;
        Button_Select_Salida[11]:=FALSE;
        Button_Select_Salida[12]:=FALSE;
        Button_Select_Salida[13]:=FALSE;
        Button_Select_Salida[14]:=FALSE;
        Button_Select_Salida[15]:=FALSE;
        Button_Select_Salida[16]:=FALSE;
        END_IF

IF EDGEPOS(Button_Select_Salida[8]) THEN
        Button_Select_Salida[1]:=FALSE;
        Button_Select_Salida[2]:=FALSE;
        Button_Select_Salida[3]:=FALSE;
        Button_Select_Salida[4]:=FALSE;
        Button_Select_Salida[5]:=FALSE;
        Button_Select_Salida[6]:=FALSE;
        Button_Select_Salida[7]:=FALSE;
        Button_Select_Salida[9]:=FALSE;
        Button_Select_Salida[10]:=FALSE;
        Button_Select_Salida[11]:=FALSE;
        Button_Select_Salida[12]:=FALSE;
        Button_Select_Salida[13]:=FALSE;
        Button_Select_Salida[14]:=FALSE;
        Button_Select_Salida[15]:=FALSE;
        Button_Select_Salida[16]:=FALSE;
        END_IF

IF EDGEPOS(Button_Select_Salida[9]) THEN
        Button_Select_Salida[1]:=FALSE;
        Button_Select_Salida[2]:=FALSE;
        Button_Select_Salida[3]:=FALSE;
        Button_Select_Salida[4]:=FALSE;
        Button_Select_Salida[5]:=FALSE;
        Button_Select_Salida[6]:=FALSE;
        Button_Select_Salida[7]:=FALSE;
        Button_Select_Salida[8]:=FALSE;
        Button_Select_Salida[10]:=FALSE;
        Button_Select_Salida[11]:=FALSE;
        Button_Select_Salida[12]:=FALSE;
        Button_Select_Salida[13]:=FALSE;
        Button_Select_Salida[14]:=FALSE;
        Button_Select_Salida[15]:=FALSE;
        Button_Select_Salida[16]:=FALSE;
        END_IF

IF EDGEPOS(Button_Select_Salida[10]) THEN
        Button_Select_Salida[1]:=FALSE;
        Button_Select_Salida[2]:=FALSE;
```

```
        Button_Select_Salida[3]:=FALSE;
        Button_Select_Salida[4]:=FALSE;
        Button_Select_Salida[5]:=FALSE;
        Button_Select_Salida[6]:=FALSE;
        Button_Select_Salida[7]:=FALSE;
        Button_Select_Salida[8]:=FALSE;
        Button_Select_Salida[9]:=FALSE;
        Button_Select_Salida[11]:=FALSE;
        Button_Select_Salida[12]:=FALSE;
        Button_Select_Salida[13]:=FALSE;
        Button_Select_Salida[14]:=FALSE;
        Button_Select_Salida[15]:=FALSE;
        Button_Select_Salida[16]:=FALSE;
        END_IF

IF EDGEPOS(Button_Select_Salida[11]) THEN
        Button_Select_Salida[1]:=FALSE;
        Button_Select_Salida[2]:=FALSE;
        Button_Select_Salida[3]:=FALSE;
        Button_Select_Salida[4]:=FALSE;
        Button_Select_Salida[5]:=FALSE;
        Button_Select_Salida[6]:=FALSE;
        Button_Select_Salida[7]:=FALSE;
        Button_Select_Salida[8]:=FALSE;
        Button_Select_Salida[9]:=FALSE;
        Button_Select_Salida[10]:=FALSE;
        Button_Select_Salida[12]:=FALSE;
        Button_Select_Salida[13]:=FALSE;
        Button_Select_Salida[14]:=FALSE;
        Button_Select_Salida[15]:=FALSE;
        Button_Select_Salida[16]:=FALSE;
        END_IF

IF EDGEPOS(Button_Select_Salida[12]) THEN
        Button_Select_Salida[1]:=FALSE;
        Button_Select_Salida[2]:=FALSE;
        Button_Select_Salida[3]:=FALSE;
        Button_Select_Salida[4]:=FALSE;
        Button_Select_Salida[5]:=FALSE;
        Button_Select_Salida[6]:=FALSE;
        Button_Select_Salida[7]:=FALSE;
        Button_Select_Salida[8]:=FALSE;
        Button_Select_Salida[9]:=FALSE;
        Button_Select_Salida[10]:=FALSE;
        Button_Select_Salida[11]:=FALSE;
        Button_Select_Salida[13]:=FALSE;
        Button_Select_Salida[14]:=FALSE;
        Button_Select_Salida[15]:=FALSE;
        Button_Select_Salida[16]:=FALSE;
```

```
END_IF

IF EDGEPOS(Button_Select_Salida[13]) THEN
Button_Select_Salida[1]:=FALSE;
Button_Select_Salida[2]:=FALSE;
Button_Select_Salida[3]:=FALSE;
Button_Select_Salida[4]:=FALSE;
Button_Select_Salida[5]:=FALSE;
Button_Select_Salida[6]:=FALSE;
Button_Select_Salida[7]:=FALSE;
Button_Select_Salida[8]:=FALSE;
Button_Select_Salida[9]:=FALSE;
Button_Select_Salida[10]:=FALSE;
Button_Select_Salida[11]:=FALSE;
Button_Select_Salida[12]:=FALSE;
Button_Select_Salida[14]:=FALSE;
Button_Select_Salida[15]:=FALSE;
Button_Select_Salida[16]:=FALSE;
END_IF

IF EDGEPOS(Button_Select_Salida[14]) THEN
Button_Select_Salida[1]:=FALSE;
Button_Select_Salida[2]:=FALSE;
Button_Select_Salida[3]:=FALSE;
Button_Select_Salida[4]:=FALSE;
Button_Select_Salida[5]:=FALSE;
Button_Select_Salida[6]:=FALSE;
Button_Select_Salida[7]:=FALSE;
Button_Select_Salida[8]:=FALSE;
Button_Select_Salida[9]:=FALSE;
Button_Select_Salida[10]:=FALSE;
Button_Select_Salida[11]:=FALSE;
Button_Select_Salida[12]:=FALSE;
Button_Select_Salida[13]:=FALSE;
Button_Select_Salida[15]:=FALSE;
Button_Select_Salida[16]:=FALSE;
END_IF

IF EDGEPOS(Button_Select_Salida[15]) THEN
Button_Select_Salida[1]:=FALSE;
Button_Select_Salida[2]:=FALSE;
Button_Select_Salida[3]:=FALSE;
Button_Select_Salida[4]:=FALSE;
Button_Select_Salida[5]:=FALSE;
Button_Select_Salida[6]:=FALSE;
Button_Select_Salida[7]:=FALSE;
Button_Select_Salida[8]:=FALSE;
Button_Select_Salida[9]:=FALSE;
Button_Select_Salida[10]:=FALSE;
```

```
            Button_Select_Salida[11]:=FALSE;
            Button_Select_Salida[12]:=FALSE;
            Button_Select_Salida[13]:=FALSE;
            Button_Select_Salida[14]:=FALSE;
            Button_Select_Salida[16]:=FALSE;
            END_IF
(*---- PROGRAMACIÓN DE COLORES PARA CADA LETRA SEGÚN LO SELECCIONADO POR EL OPERARIO---*)
// Para boton de salida 1
 IF Button_Select_Salida[1]=TRUE THEN
            Indice_Numero_Salida:=1;
            Salida_Activada[1]:=50; //Color rojo
 ELSE Salida_Activada[1]:=70; //Color negro
 END_IF
// Para boton de salida 2
            IF Button_Select_Salida[2] THEN
            Indice_Numero_Salida:=2;
            Salida_Activada[2]:=50; //Color rojo
 ELSE Salida_Activada[2]:=70; //Color negro
 END_IF
// Para boton de salida 3
            IF Button_Select_Salida[3]=TRUE THEN
            Indice_Numero_Salida:=3;
            Salida_Activada[3]:=50; //Color rojo
 ELSE Salida_Activada[3]:=28; //Color negro
 END_IF
// Para boton de salida 4
            IF Button_Select_Salida[4]=TRUE THEN
            Indice_Numero_Salida:=4;
            Salida_Activada[4]:=50; //Color rojo
 ELSE Salida_Activada[4]:=28; //Color negro
 END_IF
// Para boton de salida 5
            IF Button_Select_Salida[5]=TRUE THEN
            Indice_Numero_Salida:=5;
            Salida_Activada[5]:=50; //Color rojo
 ELSE Salida_Activada[5]:=70; //Color negro
 END_IF
// Para boton de salida 6
            IF Button_Select_Salida[6]=TRUE THEN
            Indice_Numero_Salida:=6;
            Salida_Activada[6]:=50; //Color rojo
 ELSE Salida_Activada[6]:=70; //Color negro
 END_IF
// Para boton de salida 7
            IF Button_Select_Salida[7]=TRUE THEN
            Indice_Numero_Salida:=7;
            Salida_Activada[7]:=50; //Color rojo
 ELSE Salida_Activada[7]:=28; //Color negro
 END_IF
```

```
// Para boton de salida 8
        IF Button_Select_Salida[8]=TRUE THEN
        Indice_Numero_Salida:=8;
        Salida_Activada[8]:=50; //Color rojo
ELSE Salida_Activada[8]:=28; //Color negro
END_IF
// Para boton de salida 9
        IF Button_Select_Salida[9]=TRUE THEN
        Indice_Numero_Salida:=9;
        Salida_Activada[9]:=50; //Color rojo
ELSE Salida_Activada[9]:=70; //Color negro
END_IF
// Para boton de salida 10
        IF Button_Select_Salida[10]=TRUE THEN
        Indice_Numero_Salida:=10;
        Salida_Activada[10]:=50; //Color rojo
ELSE Salida_Activada[10]:=70; //Color negro
END_IF
// Para boton de salida 11
        IF Button_Select_Salida[11]=TRUE THEN
        Indice_Numero_Salida:=11;
        Salida_Activada[11]:=50; //Color rojo
ELSE Salida_Activada[11]:=28; //Color negro
END_IF
// Para boton de salida 12
        IF Button_Select_Salida[12]=TRUE THEN
        Indice_Numero_Salida:=12;
        Salida_Activada[12]:=50; //Color rojo
ELSE Salida_Activada[12]:=28; //Color negro
END_IF
// Para boton de salida 13
        IF Button_Select_Salida[13]=TRUE THEN
        Indice_Numero_Salida:=13;
        Salida_Activada[13]:=50; //Color rojo
ELSE Salida_Activada[13]:=70; //Color negro
END_IF
// Para boton de salida 14
        IF Button_Select_Salida[14]=TRUE THEN
        Indice_Numero_Salida:=14;
        Salida_Activada[14]:=50; //Color rojo
ELSE Salida_Activada[14]:=70; //Color negro
END_IF
// Para boton de salida 15
        IF Button_Select_Salida[15]=TRUE THEN
        Indice_Numero_Salida:=15;
        Salida_Activada[15]:=50; //Color rojo
ELSE Salida_Activada[15]:=28; //Color negro
END_IF
```

```
(*------------------------------------ Pulsadores de seleccionar el calibre ----------------------------------*)
IF EDGEPOS(Button_Select_Calibre[1]) THEN
        Button_Select_Calibre[2]:=FALSE;
        Button_Select_Calibre[3]:=FALSE;
        Button_Select_Calibre[4]:=FALSE;
        Button_Select_Calibre[5]:=FALSE;
        Button_Select_Calibre[6]:=FALSE;
        Button_Select_Calibre[7]:=FALSE;
        Button_Select_Calibre[8]:=FALSE;
        Button_Select_Calibre[9]:=FALSE;
        Button_Select_Calibre[10]:=FALSE;
        Button_Select_Calibre[11]:=FALSE;
        Button_Select_Calibre[12]:=FALSE;
        Button_Select_Calibre[13]:=FALSE;
        Button_Select_Calibre[14]:=FALSE;
        Button_Select_Calibre[15]:=FALSE;
        Button_Select_Calibre[16]:=FALSE;
        END_IF

IF EDGEPOS(Button_Select_Calibre[2]) THEN
        Button_Select_Calibre[1]:=FALSE;
        Button_Select_Calibre[3]:=FALSE;
        Button_Select_Calibre[4]:=FALSE;
        Button_Select_Calibre[5]:=FALSE;
        Button_Select_Calibre[6]:=FALSE;
        Button_Select_Calibre[7]:=FALSE;
        Button_Select_Calibre[8]:=FALSE;
        Button_Select_Calibre[9]:=FALSE;
        Button_Select_Calibre[10]:=FALSE;
        Button_Select_Calibre[11]:=FALSE;
        Button_Select_Calibre[12]:=FALSE;
        Button_Select_Calibre[13]:=FALSE;
        Button_Select_Calibre[14]:=FALSE;
        Button_Select_Calibre[15]:=FALSE;
        Button_Select_Calibre[16]:=FALSE;
        END_IF

IF EDGEPOS(Button_Select_Calibre[3]) THEN
        Button_Select_Calibre[1]:=FALSE;
        Button_Select_Calibre[2]:=FALSE;
        Button_Select_Calibre[4]:=FALSE;
        Button_Select_Calibre[5]:=FALSE;
        Button_Select_Calibre[6]:=FALSE;
        Button_Select_Calibre[7]:=FALSE;
        Button_Select_Calibre[8]:=FALSE;
        Button_Select_Calibre[9]:=FALSE;
        Button_Select_Calibre[10]:=FALSE;
        Button_Select_Calibre[11]:=FALSE;
        Button_Select_Calibre[12]:=FALSE;
```

```
        Button_Select_Calibre[13]:=FALSE;
        Button_Select_Calibre[14]:=FALSE;
        Button_Select_Calibre[15]:=FALSE;
        Button_Select_Calibre[16]:=FALSE;
        END_IF

IF EDGEPOS(Button_Select_Calibre[4]) THEN
        Button_Select_Calibre[1]:=FALSE;
        Button_Select_Calibre[2]:=FALSE;
        Button_Select_Calibre[3]:=FALSE;
        Button_Select_Calibre[5]:=FALSE;
        Button_Select_Calibre[6]:=FALSE;
        Button_Select_Calibre[7]:=FALSE;
        Button_Select_Calibre[8]:=FALSE;
        Button_Select_Calibre[9]:=FALSE;
        Button_Select_Calibre[10]:=FALSE;
        Button_Select_Calibre[11]:=FALSE;
        Button_Select_Calibre[12]:=FALSE;
        Button_Select_Calibre[13]:=FALSE;
        Button_Select_Calibre[14]:=FALSE;
        Button_Select_Calibre[15]:=FALSE;
        Button_Select_Calibre[16]:=FALSE;
        END_IF

IF EDGEPOS(Button_Select_Calibre[5]) THEN
        Button_Select_Calibre[1]:=FALSE;
        Button_Select_Calibre[2]:=FALSE;
        Button_Select_Calibre[3]:=FALSE;
        Button_Select_Calibre[4]:=FALSE;
        Button_Select_Calibre[6]:=FALSE;
        Button_Select_Calibre[7]:=FALSE;
        Button_Select_Calibre[8]:=FALSE;
        Button_Select_Calibre[9]:=FALSE;
        Button_Select_Calibre[10]:=FALSE;
        Button_Select_Calibre[11]:=FALSE;
        Button_Select_Calibre[12]:=FALSE;
        Button_Select_Calibre[13]:=FALSE;
        Button_Select_Calibre[14]:=FALSE;
        Button_Select_Calibre[15]:=FALSE;
        Button_Select_Calibre[16]:=FALSE;
        END_IF

        IF EDGEPOS(Button_Select_Calibre[6]) THEN
        Button_Select_Calibre[1]:=FALSE;
        Button_Select_Calibre[2]:=FALSE;
        Button_Select_Calibre[3]:=FALSE;
        Button_Select_Calibre[4]:=FALSE;
        Button_Select_Calibre[5]:=FALSE;
        Button_Select_Calibre[7]:=FALSE;
```

```
Button_Select_Calibre[8]:=FALSE;
Button_Select_Calibre[9]:=FALSE;
Button_Select_Calibre[10]:=FALSE;
Button_Select_Calibre[11]:=FALSE;
Button_Select_Calibre[12]:=FALSE;
Button_Select_Calibre[13]:=FALSE;
Button_Select_Calibre[14]:=FALSE;
Button_Select_Calibre[15]:=FALSE;
Button_Select_Calibre[16]:=FALSE;
END_IF

IF EDGEPOS(Button_Select_Calibre[7]) THEN
Button_Select_Calibre[1]:=FALSE;
Button_Select_Calibre[2]:=FALSE;
Button_Select_Calibre[3]:=FALSE;
Button_Select_Calibre[4]:=FALSE;
Button_Select_Calibre[5]:=FALSE;
Button_Select_Calibre[6]:=FALSE;
Button_Select_Calibre[8]:=FALSE;
Button_Select_Calibre[9]:=FALSE;
Button_Select_Calibre[10]:=FALSE;
Button_Select_Calibre[11]:=FALSE;
Button_Select_Calibre[12]:=FALSE;
Button_Select_Calibre[13]:=FALSE;
Button_Select_Calibre[14]:=FALSE;
Button_Select_Calibre[15]:=FALSE;
Button_Select_Calibre[16]:=FALSE;
END_IF

IF EDGEPOS(Button_Select_Calibre[8]) THEN
Button_Select_Calibre[1]:=FALSE;
Button_Select_Calibre[2]:=FALSE;
Button_Select_Calibre[3]:=FALSE;
Button_Select_Calibre[4]:=FALSE;
Button_Select_Calibre[5]:=FALSE;
Button_Select_Calibre[6]:=FALSE;
Button_Select_Calibre[7]:=FALSE;
Button_Select_Calibre[9]:=FALSE;
Button_Select_Calibre[10]:=FALSE;
Button_Select_Calibre[11]:=FALSE;
Button_Select_Calibre[12]:=FALSE;
Button_Select_Calibre[13]:=FALSE;
Button_Select_Calibre[14]:=FALSE;
Button_Select_Calibre[15]:=FALSE;
Button_Select_Calibre[16]:=FALSE;
END_IF

IF EDGEPOS(Button_Select_Calibre[9]) THEN
Button_Select_Calibre[1]:=FALSE;
```

```
Button_Select_Calibre[2]:=FALSE;
Button_Select_Calibre[3]:=FALSE;
Button_Select_Calibre[4]:=FALSE;
Button_Select_Calibre[5]:=FALSE;
Button_Select_Calibre[6]:=FALSE;
Button_Select_Calibre[7]:=FALSE;
Button_Select_Calibre[8]:=FALSE;
Button_Select_Calibre[10]:=FALSE;
Button_Select_Calibre[11]:=FALSE;
Button_Select_Calibre[12]:=FALSE;
Button_Select_Calibre[13]:=FALSE;
Button_Select_Calibre[14]:=FALSE;
Button_Select_Calibre[15]:=FALSE;
Button_Select_Calibre[16]:=FALSE;
END_IF

IF EDGEPOS(Button_Select_Calibre[10]) THEN
Button_Select_Calibre[1]:=FALSE;
Button_Select_Calibre[2]:=FALSE;
Button_Select_Calibre[3]:=FALSE;
Button_Select_Calibre[4]:=FALSE;
Button_Select_Calibre[5]:=FALSE;
Button_Select_Calibre[6]:=FALSE;
Button_Select_Calibre[7]:=FALSE;
Button_Select_Calibre[8]:=FALSE;
Button_Select_Calibre[9]:=FALSE;
Button_Select_Calibre[11]:=FALSE;
Button_Select_Calibre[12]:=FALSE;
Button_Select_Calibre[13]:=FALSE;
Button_Select_Calibre[14]:=FALSE;
Button_Select_Calibre[15]:=FALSE;
Button_Select_Calibre[16]:=FALSE;
END_IF

IF EDGEPOS(Button_Select_Calibre[11]) THEN
Button_Select_Calibre[1]:=FALSE;
Button_Select_Calibre[2]:=FALSE;
Button_Select_Calibre[3]:=FALSE;
Button_Select_Calibre[4]:=FALSE;
Button_Select_Calibre[5]:=FALSE;
Button_Select_Calibre[6]:=FALSE;
Button_Select_Calibre[7]:=FALSE;
Button_Select_Calibre[8]:=FALSE;
Button_Select_Calibre[9]:=FALSE;
Button_Select_Calibre[10]:=FALSE;
Button_Select_Calibre[12]:=FALSE;
Button_Select_Calibre[13]:=FALSE;
Button_Select_Calibre[14]:=FALSE;
Button_Select_Calibre[15]:=FALSE;
```

```
Button_Select_Calibre[16]:=FALSE;
END_IF

IF EDGEPOS(Button_Select_Calibre[12]) THEN
Button_Select_Calibre[1]:=FALSE;
Button_Select_Calibre[2]:=FALSE;
Button_Select_Calibre[3]:=FALSE;
Button_Select_Calibre[4]:=FALSE;
Button_Select_Calibre[5]:=FALSE;
Button_Select_Calibre[6]:=FALSE;
Button_Select_Calibre[7]:=FALSE;
Button_Select_Calibre[8]:=FALSE;
Button_Select_Calibre[9]:=FALSE;
Button_Select_Calibre[10]:=FALSE;
Button_Select_Calibre[11]:=FALSE;
Button_Select_Calibre[13]:=FALSE;
Button_Select_Calibre[14]:=FALSE;
Button_Select_Calibre[15]:=FALSE;
Button_Select_Calibre[16]:=FALSE;
END_IF

IF EDGEPOS(Button_Select_Calibre[13]) THEN
Button_Select_Calibre[1]:=FALSE;
Button_Select_Calibre[2]:=FALSE;
Button_Select_Calibre[3]:=FALSE;
Button_Select_Calibre[4]:=FALSE;
Button_Select_Calibre[5]:=FALSE;
Button_Select_Calibre[6]:=FALSE;
Button_Select_Calibre[7]:=FALSE;
Button_Select_Calibre[8]:=FALSE;
Button_Select_Calibre[9]:=FALSE;
Button_Select_Calibre[10]:=FALSE;
Button_Select_Calibre[11]:=FALSE;
Button_Select_Calibre[12]:=FALSE;
Button_Select_Calibre[14]:=FALSE;
Button_Select_Calibre[15]:=FALSE;
Button_Select_Calibre[16]:=FALSE;
END_IF

IF EDGEPOS(Button_Select_Calibre[14]) THEN
Button_Select_Calibre[1]:=FALSE;
Button_Select_Calibre[2]:=FALSE;
Button_Select_Calibre[3]:=FALSE;
Button_Select_Calibre[4]:=FALSE;
Button_Select_Calibre[5]:=FALSE;
Button_Select_Calibre[6]:=FALSE;
Button_Select_Calibre[7]:=FALSE;
Button_Select_Calibre[8]:=FALSE;
Button_Select_Calibre[9]:=FALSE;
```

```
Button_Select_Calibre[10]:=FALSE;
Button_Select_Calibre[11]:=FALSE;
Button_Select_Calibre[12]:=FALSE;
Button_Select_Calibre[13]:=FALSE;
Button_Select_Calibre[15]:=FALSE;
Button_Select_Calibre[16]:=FALSE;
END_IF

IF EDGEPOS(Button_Select_Calibre[15]) THEN
Button_Select_Calibre[1]:=FALSE;
Button_Select_Calibre[2]:=FALSE;
Button_Select_Calibre[3]:=FALSE;
Button_Select_Calibre[4]:=FALSE;
Button_Select_Calibre[5]:=FALSE;
Button_Select_Calibre[6]:=FALSE;
Button_Select_Calibre[7]:=FALSE;
Button_Select_Calibre[8]:=FALSE;
Button_Select_Calibre[9]:=FALSE;
Button_Select_Calibre[10]:=FALSE;
Button_Select_Calibre[11]:=FALSE;
Button_Select_Calibre[12]:=FALSE;
Button_Select_Calibre[13]:=FALSE;
Button_Select_Calibre[14]:=FALSE;
Button_Select_Calibre[16]:=FALSE;
END_IF

IF EDGEPOS(Button_Select_Calibre[16]) THEN
Button_Select_Calibre[1]:=FALSE;
Button_Select_Calibre[2]:=FALSE;
Button_Select_Calibre[3]:=FALSE;
Button_Select_Calibre[4]:=FALSE;
Button_Select_Calibre[5]:=FALSE;
Button_Select_Calibre[6]:=FALSE;
Button_Select_Calibre[7]:=FALSE;
Button_Select_Calibre[8]:=FALSE;
Button_Select_Calibre[9]:=FALSE;
Button_Select_Calibre[10]:=FALSE;
Button_Select_Calibre[11]:=FALSE;
Button_Select_Calibre[12]:=FALSE;
Button_Select_Calibre[13]:=FALSE;
Button_Select_Calibre[14]:=FALSE;
Button_Select_Calibre[15]:=FALSE;
END_IF
// Para boton de Calibre 1
IF Button_Select_Calibre[1]=TRUE THEN
Indice_Numero_Calibre:=1;
Calibre_Activado[1]:=46; //Color morado
ELSE Calibre_Activado[1]:=88; //Color negro
END_IF
```

```
// Para boton de Calibre 2
 IF Button_Select_Calibre[2]=TRUE THEN
            Indice_Numero_Calibre:=2;
            Calibre_Activado[2]:=46; //Color morado
 ELSE Calibre_Activado[2]:=88; //Color negro
 END_IF
// Para boton de Calibre 3
 IF Button_Select_Calibre[3]=TRUE THEN
            Indice_Numero_Calibre:=3;
            Calibre_Activado[3]:=46; //Color morado
 ELSE Calibre_Activado[3]:=88; //Color negro
 END_IF
 // Para boton de Calibre 4
 IF Button_Select_Calibre[4]=TRUE THEN
            Indice_Numero_Calibre:=4;
            Calibre_Activado[4]:=46; //Color morado
 ELSE Calibre_Activado[4]:=88; //Color negro
 END_IF
// Para boton de Calibre 5
 IF Button_Select_Calibre[5]=TRUE THEN
            Indice_Numero_Calibre:=5;
            Calibre_Activado[5]:=46; //Color morado
 ELSE Calibre_Activado[5]:=88; //Color negro
 END_IF
 // Para boton de Calibre 6
 IF Button_Select_Calibre[6]=TRUE THEN
            Indice_Numero_Calibre:=6;
            Calibre_Activado[6]:=46; //Color morado
 ELSE Calibre_Activado[6]:=88; //Color negro
 END_IF
// Para boton de Calibre 7
 IF Button_Select_Calibre[7]=TRUE THEN
            Indice_Numero_Calibre:=7;
            Calibre_Activado[7]:=46; //Color morado
 ELSE Calibre_Activado[7]:=88; //Color negro
 END_IF
// Para boton de Calibre 8
 IF Button_Select_Calibre[8]=TRUE THEN
            Indice_Numero_Calibre:=8;
            Calibre_Activado[8]:=46; //Color morado
 ELSE Calibre_Activado[8]:=88; //Color negro
 END_IF
// Para boton de Calibre 9
 IF Button_Select_Calibre[9]=TRUE THEN
            Indice_Numero_Calibre:=9;
            Calibre_Activado[9]:=46; //Color morado
 ELSE Calibre_Activado[9]:=88; //Color negro
 END_IF
```

```
// Para boton de Calibre 10
 IF Button_Select_Calibre[10]=TRUE THEN
            Indice_Numero_Calibre:=10;
            Calibre_Activado[10]:=46; //Color morado
 ELSE Calibre_Activado[10]:=88; //Color negro
 END_IF
// Para boton de Calibre 11
 IF Button_Select_Calibre[11]=TRUE THEN
            Indice_Numero_Calibre:=11;
            Calibre_Activado[11]:=46; //Color morado
 ELSE Calibre_Activado[11]:=88; //Color negro
 END_IF
// Para boton de Calibre 12
 IF Button_Select_Calibre[12]=TRUE THEN
            Indice_Numero_Calibre:=12;
            Calibre_Activado[12]:=46; //Color morado
 ELSE Calibre_Activado[12]:=88; //Color negro
 END_IF
// Para boton de Calibre 13
 IF Button_Select_Calibre[13]=TRUE THEN
            Indice_Numero_Calibre:=13;
            Calibre_Activado[13]:=46; //Color morado
 ELSE Calibre_Activado[13]:=88; //Color negro
 END_IF
// Para boton de Calibre 14
 IF Button_Select_Calibre[14]=TRUE THEN
            Indice_Numero_Calibre:=14;
            Calibre_Activado[14]:=46; //Color morado
 ELSE Calibre_Activado[14]:=88; //Color negro
 END_IF
// Para boton de Calibre 15
 IF Button_Select_Calibre[15]=TRUE THEN
            Indice_Numero_Calibre:=15;
            Calibre_Activado[15]:=46; //Color morado
 ELSE Calibre_Activado[15]:=88; //Color negro
 END_IF
// Para boton de Calibre 16
 IF Button_Select_Calibre[16]=TRUE THEN
            Indice_Numero_Calibre:=16;
            Calibre_Activado[16]:=46; //Color morado
 ELSE Calibre_Activado[16]:=88; //Color negro
 END_IF

(*---------------------------------------------------------- ASIGNAR SALIDAS ----------------------------------------------------------*)
 IF EDGEPOS(Button_Asignar_Salida) THEN
            Asignar_Salida:=TRUE;
            Valor_Existe:=0;
            Valor_Guardado:=0;
 END_IF
```

```
IF Asignar_Salida=TRUE THEN
        FOR g:=1 TO 20 DO
//comprobamos que el valor no esté ya asignado en la salida
          IF Layaut_Salida[Indice_Numero_Salida].Calibre[g]=Indice_Numero_Calibre THEN

//en el caso de que calibremos solo por peso o por diámetro
                IF (Pantalla_Layaut.Modo_de_Calibracion=0 OR Pantalla_Layaut.Modo_de_Calibracion=3) THEN
                  Valor_Existe:=TRUE;
                END_IF
            FOR h:=1 TO 20 DO
              IF Layaut_Salida[Indice_Numero_Salida].Grado[h]=Grado[Indice_Numero_Grado] THEN
              Valor_Existe:=TRUE;
              END_IF
            END_FOR
    END_IF
        END_FOR

  FOR k:=1 TO 20 DO
//si el valor no existe en la lista entonces lo asignamos
          IF (Layaut_Salida[Indice_Numero_Salida].Calibre[k]=0) AND Valor_Existe=FALSE AND
Valor_Guardado=FALSE THEN
                        Layaut_Salida[Indice_Numero_Salida].Calibre[k]:=Indice_Numero_Calibre;
                        Layaut_Salida[Indice_Numero_Salida].Grado[k]:=Grado[Indice_Numero_Grado];
                        Valor_Guardado:=TRUE;
                END_IF
        END_FOR
END_IF
IF Valor_Guardado=TRUE OR Valor_Existe=TRUE THEN
        Asignar_Salida:=FALSE;
        END_IF

(*---------------------------------------------------- BORRAR TODA LA SALIDA ----------------------------------------------------*)
IF EDGEPOS(Button_Borrar_Todo) THEN
        FOR m:=1 TO 20 DO
        Layaut_Salida[Indice_Numero_Salida].Calibre[m]:=0;
        Layaut_Salida[Indice_Numero_Salida].Grado[m]:=' ';
        END_FOR
END_IF

IF Valor_Existe=1 THEN
        Pantalla_Emerg_Valor_Existente:=0;
        ELSE
        Pantalla_Emerg_Valor_Existente:=1;
END_IF

(*---------------------------------------------------- BORRAR DE LA SALIDA ----------------------------------------------------*)
IF EDGEPOS(Button_Borrar_Salida) THEN
        Borrar_de_Salida:=TRUE;
        Borrado_de_Salida:=FALSE;
```

```
            Salida_no_Asignada:=FALSE;
 END_IF

 IF Borrar_de_Salida=TRUE THEN
            FOR a:=1 TO 20 DO
               IF (Layaut_Salida[Indice_Numero_Salida].Calibre[a]=Indice_Numero_Calibre AND
Borrado_de_Salida=FALSE) THEN

                  FOR b:=1 TO 20 DO
                     IF Layaut_Salida[Indice_Numero_Salida].Grado[b]=Grado[Indice_Numero_Grado] AND
Borrado_de_Salida=FALSE THEN
                     Layaut_Salida[Indice_Numero_Salida].Calibre[a]:=0;
                                    Layaut_Salida[Indice_Numero_Salida].Grado[a]:=' ';
                                    Borrado_de_Salida:=TRUE;
                     END_IF
                  END_FOR
               END_IF
    END_FOR

 END_IF

 IF Borrado_de_Salida=TRUE OR Salida_no_Asignada=TRUE THEN
            Borrar_de_Salida:=FALSE;
 END_IF

 IF Borrar_de_Salida=TRUE AND Borrado_de_Salida=FALSE THEN
            Salida_no_Asignada:=TRUE;
 END_IF

 IF Salida_no_Asignada=TRUE THEN
            Pantalla_Emerg_Salida_no_Asig:=0;
            ELSE
            Pantalla_Emerg_Salida_no_Asig:=1;
 END_IF

(*---------- PASAR TODAS LOS CALIBRES ASIGNADOS A STRING PARA VISUALIZAR EN PANTALLA ----------*)
 IF EDGEPOS(Button_Asignar_Salida) OR EDGEPOS(Button_Borrar_Salida) OR EDGEPOS(Button_Borrar_Todo)
THEN
            strcpy(ADR(strSalida[Indice_Numero_Salida]),ADR(''));
            FOR i:=1 TO 20 DO
            IF Layaut_Salida[Indice_Numero_Salida].Calibre[i]<>0 THEN

strcat(ADR(strSalida[Indice_Numero_Salida]),ADR(Variedad_de_la_Fruta_Lectura.Nombre_del_Grado[Layaut_Salida[Ind
ice_Numero_Salida].Calibre[i]]));
                                    IF (Pantalla_Layaut.Modo_de_Calibracion=1 OR
Pantalla_Layaut.Modo_de_Calibracion=2 OR Pantalla_Layaut.Modo_de_Calibracion=4 OR
Pantalla_Layaut.Modo_de_Calibracion=5) THEN
```

```
// Modo de calibración: 1=Peso y color, 2=Peso color y diámetro, 4=Diámetro y color, y 5= solo color)
                                    strcat(ADR(strSalida[Indice_Numero_Salida]),ADR('/'));

strcat(ADR(strSalida[Indice_Numero_Salida]),ADR(Layaut_Salida[Indice_Numero_Salida].Grado[i]));
                                    END_IF
                                    strcat(ADR(strSalida[Indice_Numero_Salida]),ADR(';'));
                            END_IF
    END_FOR
 END_IF
(*-- bloqueamos la opción de la pantalla de layaut según lo que tenemos configurado en el modo de calibración ---------*)
// bloquemaos la opción de grado si solo tenemos peso o diámetro
 IF (Pantalla_Layaut.Modo_de_Calibracion=0 OR Pantalla_Layaut.Modo_de_Calibracion=3) THEN
   Bloquear_grado:=1;
            ELSE
            Bloquear_grado:=0;
 END_IF

 IF (Pantalla_Layaut.Modo_de_Calibracion=0 OR Pantalla_Layaut.Modo_de_Calibracion=1 OR
Pantalla_Layaut.Modo_de_Calibracion=5) THEN
   Bloquear_diametro:=0;
            ELSE
            Bloquear_diametro:=1;
 END_IF

 IF (Pantalla_Layaut.Modo_de_Calibracion=3 OR Pantalla_Layaut.Modo_de_Calibracion=4 OR
Pantalla_Layaut.Modo_de_Calibracion=5) THEN
   Bloquear_peso:=0;
            ELSE
            Bloquear_peso:=1;
 END_IF

(*---------------------- MEMORIZAR LOS DATOS DE LA PANTALLA LAYAUT EN LA MEMORIA ----------------------------*)
 IF EDGEPOS(Button_Layaut_UP) THEN
            Numero_Layaut:=Numero_Layaut+1;
            Leer_Layaut:=TRUE;
            ELSIF EDGEPOS(Button_Layaut_DOWN) THEN
            Numero_Layaut:=Numero_Layaut-1;
            Leer_Layaut:=TRUE;
            ELSIF EDGEPOS(Button_Leer_Layaut) THEN
            Leer_Layaut:=TRUE;
 END_IF

 IF EDGEPOS(Button_Guardar_Layaut) THEN
            Pantalla_Emerg_Guardar_Layaut:=1;
            Guardar_Layaut:=TRUE;
 END_IF
```

```
(*---------------------------------- GUARDAR LAYAUT EN MEMORIA ----------------------------------*)
 IF Guardar_Layaut=TRUE THEN

        Pantalla_Layaut_Memoria[Numero_Layaut].Nombre_Layaut:=Pantalla_Layaut.Nombre_Layaut;
        Pantalla_Layaut_Memoria[Numero_Layaut].Modo_de_Calibracion:=Pantalla_Layaut.Modo_de_Calibracion;
        Pantalla_Layaut_Memoria[Numero_Layaut].Numero_de_Variedad:=Numero_Variedad;
        Pantalla_Layaut_Memoria[Numero_Layaut].Numero_de_Grado:=Numero_Grado;

        FOR j:=1 TO (Config_Numero_Salidas + 1) DO //Para todas las salidas

                Pantalla_Layaut_Memoria[Numero_Layaut].Salidas_Asignadas[j].Conteo:=Layaut_Salida[j].Conteo;

                FOR i:=1 TO 20 DO //Para la posición de cada salida

            Pantalla_Layaut_Memoria[Numero_Layaut].Salidas_Asignadas[j].Calibre[i]:=Layaut_Salida[j].Calibre[i];

        Pantalla_Layaut_Memoria[Numero_Layaut].Salidas_Asignadas[j].Grado[i]:=Layaut_Salida[j].Grado[i];
                END_FOR
   END_FOR
        Guardar_Layaut:=FALSE;
 END_IF
(*---------------------------------- LEER EL LAYAUT DE LA MEMORIA ----------------------------------*)
 IF Leer_Layaut=TRUE THEN
        Numero_Variedad:=Pantalla_Layaut_Memoria[Numero_Layaut].Numero_de_Variedad;
        Variedad_de_la_Fruta_Lectura:=Variedad_de_la_Fruta_Memoria[Numero_Variedad];

        Numero_Grado:=Pantalla_Layaut_Memoria[Numero_Layaut].Numero_de_Grado;
        Grado_de_la_Fruta_Lectura:=Grado_de_la_Fruta_Memoria[Numero_Grado];

        Pantalla_Layaut.Nombre_Layaut:=Pantalla_Layaut_Memoria[Numero_Layaut].Nombre_Layaut;
        Pantalla_Layaut.Modo_de_Calibracion:=Pantalla_Layaut_Memoria[Numero_Layaut].Modo_de_Calibracion;

        FOR j:=1 TO (Config_Numero_Salidas + 1) DO //Para todas las salidas

                Layaut_Salida[j].Conteo:=Pantalla_Layaut_Memoria[Numero_Layaut].Salidas_Asignadas[j].Conteo;

                FOR i:=1 TO 20 DO //Para la posición de cada salida

            Layaut_Salida[j].Calibre[i]:=Pantalla_Layaut_Memoria[Numero_Layaut].Salidas_Asignadas[j].Calibre[i];

        Layaut_Salida[j].Grado[i]:=Pantalla_Layaut_Memoria[Numero_Layaut].Salidas_Asignadas[j].Grado[i];
                END_FOR
   END_FOR

//   Después de leer todos los datos de la memoria escribimos los strings de las salidas para la visualización
        FOR j:=1 TO (Config_Numero_Salidas + 1) DO //Para todas las salidas

                strcpy(ADR(strSalida[j]),ADR('')); //Ponemos a cero cada salida antes de escribir el valor de las
salidas
```

```
            FOR i:=1 TO 20 DO //Para la posición de cada salida

              IF Layaut_Salida[j].Calibre[i]<>0 THEN

strcat(ADR(strSalida[j]),ADR(Variedad_de_la_Fruta_Lectura.Nombre_del_Grado[Layaut_Salida[j].Calibre[i]]));
// si el modo de calibración también es con grado entonces también leemos el grado
                                IF (Pantalla_Layaut.Modo_de_Calibracion=1 OR
Pantalla_Layaut.Modo_de_Calibracion=2 OR Pantalla_Layaut.Modo_de_Calibracion=4 OR
Pantalla_Layaut.Modo_de_Calibracion=5) THEN
//                                Modo de calibración: 1=Peso y color, 2=Peso color y diámetro, 4=Diámetro y color, y 5=
solo color, )
                                  strcat(ADR(strSalida[j]),ADR('/'));
                                  strcat(ADR(strSalida[j]),ADR(Layaut_Salida[j].Grado[i]));
                                END_IF
                                strcat(ADR(strSalida[j]),ADR(';'));

                              END_IF
            END_FOR
          END_FOR
          Leer_Layaut:=FALSE;
END_IF

(*------------------------ COMPROVAR SI LOS CALIBRES TIENEN ALGUNA SALIDA ASIGNADA ----------------------*)
 IF EDGEPOS(Button_Asignar_Salida) OR EDGEPOS(Button_Borrar_Salida) OR EDGEPOS(Button_Borrar_Todo) OR
EDGEPOS(Button_Leer_Layaut) OR EDGEPOS(Button_Layaut_UP) OR EDGEPOS(Button_Layaut_DOWN) THEN

  FOR j:=0 TO 16 DO // Para cada calibre
    Calibre_Asignado[j]:=0;
        END_FOR
        FOR j:=1 TO (Config_Numero_Salidas + 1) DO //Para número de salidas
          FOR i:=1 TO 20 DO //Para cada posición de cada salida

                  IF Layaut_Salida[j].Calibre[i] = 1  THEN
                    Calibre_Asignado[1] := Calibre_Asignado[1] + 1;
                        ELSIF Layaut_Salida[j].Calibre[i] = 2  THEN
                    Calibre_Asignado[2] := Calibre_Asignado[2] + 1;
                        ELSIF Layaut_Salida[j].Calibre[i] = 3  THEN
                  Calibre_Asignado[3] := Calibre_Asignado[3] + 1;
                        ELSIF Layaut_Salida[j].Calibre[i] = 4  THEN
                    Calibre_Asignado[4] := Calibre_Asignado[4] + 1;
            ELSIF Layaut_Salida[j].Calibre[i] = 5  THEN
                    Calibre_Asignado[5] := Calibre_Asignado[5] + 1;
            ELSIF Layaut_Salida[j].Calibre[i] = 6  THEN
                    Calibre_Asignado[6] := Calibre_Asignado[6] + 1;
            ELSIF Layaut_Salida[j].Calibre[i] = 7  THEN
                    Calibre_Asignado[7] := Calibre_Asignado[7] + 1;
            ELSIF Layaut_Salida[j].Calibre[i] = 8  THEN
                    Calibre_Asignado[8] := Calibre_Asignado[8] + 1;
                        ELSIF Layaut_Salida[j].Calibre[i] = 9  THEN
```

```
                    Calibre_Asignado[9] := Calibre_Asignado[9] + 1;
                        ELSIF Layaut_Salida[j].Calibre[i] = 10  THEN
                    Calibre_Asignado[10] := Calibre_Asignado[10] + 1;
                        ELSIF Layaut_Salida[j].Calibre[i] = 11  THEN
                    Calibre_Asignado[11] := Calibre_Asignado[11] + 1;
                        ELSIF Layaut_Salida[j].Calibre[i] = 12  THEN
                    Calibre_Asignado[12] := Calibre_Asignado[12] + 1;
                        ELSIF Layaut_Salida[j].Calibre[i] = 13  THEN
                    Calibre_Asignado[13] := Calibre_Asignado[13] + 1;
                        ELSIF Layaut_Salida[j].Calibre[i] = 14  THEN
                    Calibre_Asignado[14] := Calibre_Asignado[14] + 1;
                        ELSIF Layaut_Salida[j].Calibre[i] = 15  THEN
                    Calibre_Asignado[15] := Calibre_Asignado[15] + 1;
                        ELSIF Layaut_Salida[j].Calibre[i] = 16  THEN
                    Calibre_Asignado[16] := Calibre_Asignado[16] + 1;
                END_IF
              END_FOR
            END_FOR

            FOR j:=1 TO 16 DO
              IF    Calibre_Asignado[j]>0 THEN //Si no hemos encontrado el calibre en ninguna posición (de 0 a 20) en ninguna salida, entonces cambiamos el texto de color
                        Color_Calibre_Asignado[j]:=15; //fondo de color blanco (significa salida asignada)
                    ELSIF Calibre_Asignado[j]=0 AND Variedad_de_la_Fruta_Lectura.Numero_de_Grados >= j THEN
                        Color_Calibre_Asignado[j]:=50; //Texto de la pantalla del calibre de color rojo (significa que no tiene ninguna salida asignada). Tambien se verifica en la rutina de alarmas
                    END_IF
            END_FOR
 END_IF
END_PROGRAM
```

Así pues, la pantalla de configuración de asignación de salidas del operario quedaría del siguiente modo:

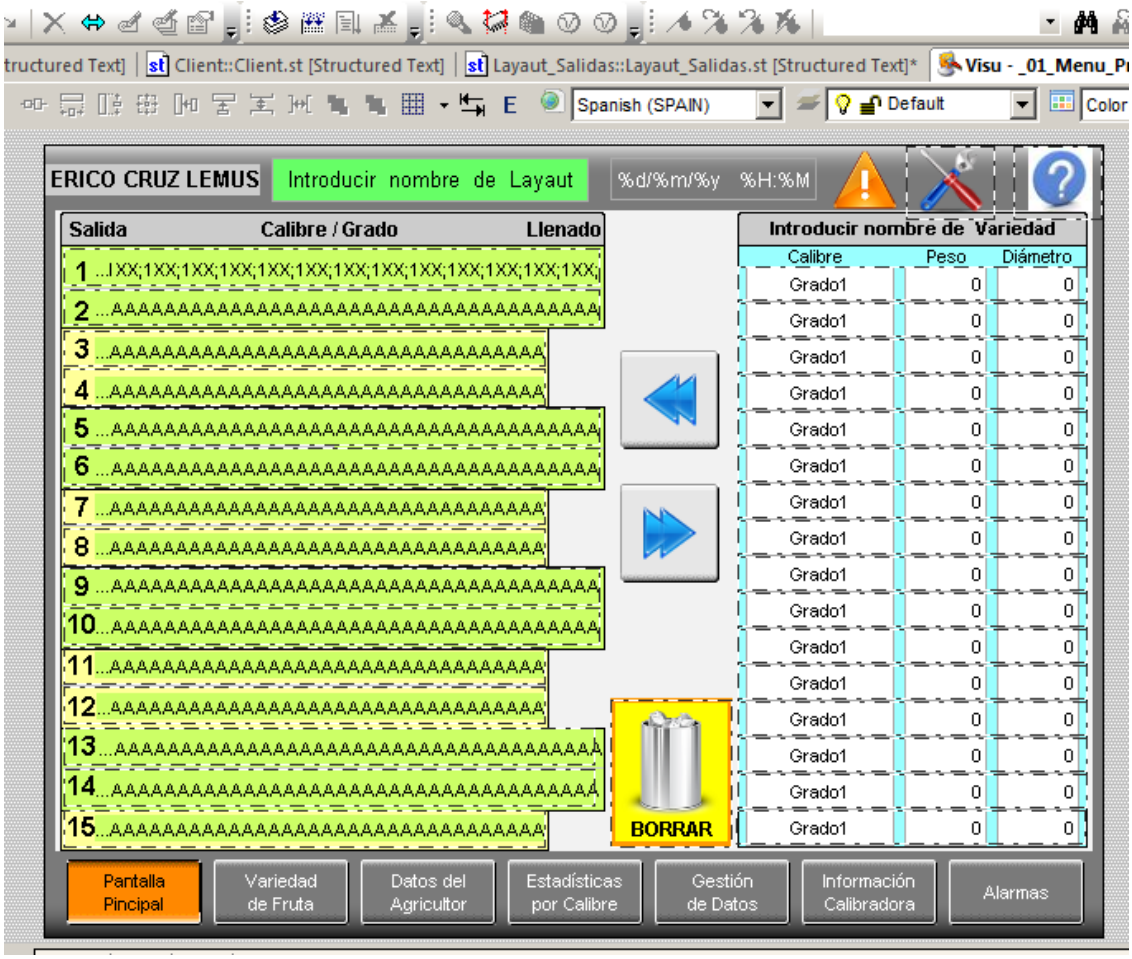


Ilustración 54

4.3.7. Alarmas

Con el módulo de alarmas gestionamos todas las alarmas que notificaremos por la pantalla táctil a operario e incluso según que alarma sea pararemos la calibradora a través de una salida digital del PLC y un relé conectado al variador de frecuencia del motor principal de la calibradora.

```
(*********************************************************************
 * COPYRIGHT --
 *********************************************************************
 * Program: Alarms
 * File: Alarms.st
 * Author: Erico Cruz
 * Created: March 20, 2010
 *********************************************************************
 * Implementation of program Alarms
 *********************************************************************)
PROGRAM _INIT
(* TODO : Add your code here *)
END_PROGRAM
```

```
PROGRAM _CYCLIC
// comprovar si hay alguna alarma activada y parpadear un aviso en la pantalla
 Alarma_Activada:=1;
  FOR i:=0 TO 50 DO
            IF Alarma[i]=TRUE THEN
                    IF Clock_1seg=TRUE THEN
                      Alarma_Activada:=0;
                            ELSE
                      Alarma_Activada:=1;
                    END_IF
            END_IF
  END_FOR

// Alarma [0], fallo magnetotérmico de electroimanes
 IF DI2_Magneto_Electroimanes=FALSE THEN
            Alarma[0]:=TRUE;
   END_IF

// Alarma [1] modo de calibración con máquina en funcionamiento
 IF EDGEPOS (gControl.User.di_Trigger) AND (Config_Modo_Calibracion=1 OR Config_Modo_Calibracion=2) THEN
            Alarma[1]:=TRUE;
   ELSIF Config_Modo_Calibracion=0 THEN
            Alarma[1]:=FALSE;
   END_IF

// Alarma [2] cámaras en estado de error
 IF DI4_Camaras_Estado_Error=TRUE THEN
            Alarma[2]:=FALSE;
   END_IF

// Alarma [4] calibres sin salida asignada
 IF EDGEPOS(gControl.User.di_Trigger) AND Alarma[4]=FALSE THEN
            FOR k:=1 TO Variedad_de_la_Fruta_Lectura.Numero_de_Grados DO
               IF Color_Calibre_Asignado[k] = 50 THEN
                  Alarma[4]:=TRUE;
               END_IF
            END_FOR
 END_IF

// Alarma [5] paso cazoletas en línea 1
 IF (EDGEPOS(gControl.User.di_Trigger) AND Calculos_Linea[1].Peso_Actual < Config_Peso_Minimo_Cazoleta AND
Config_Numero_Lineas >= 1) THEN
            Alarma[5]:=TRUE;
            END_IF

// Alarma [6] paso cazoletas en línea 2
 IF (EDGEPOS(gControl.User.di_Trigger) AND Calculos_Linea[2].Peso_Actual < Config_Peso_Minimo_Cazoleta AND
Config_Numero_Lineas >= 2) THEN
            Alarma[6]:=TRUE;
```

```
            END_IF

// Alarma [7] paso cazoletas en línea 3
 IF (EDGEPOS(gControl.User.di_Trigger) AND Calculos_Linea[3].Peso_Actual < Config_Peso_Minimo_Cazoleta AND
Config_Numero_Lineas >= 3) THEN
            Alarma[7]:=TRUE;
            END_IF

// Alarma [8] paso cazoletas en línea 4
 IF (EDGEPOS(gControl.User.di_Trigger) AND Calculos_Linea[4].Peso_Actual < Config_Peso_Minimo_Cazoleta AND
Config_Numero_Lineas >= 4) THEN
            Alarma[8]:=TRUE;
            END_IF

// Alarma [9] paso cazoletas en línea 5
 IF (EDGEPOS(gControl.User.di_Trigger) AND Calculos_Linea[5].Peso_Actual < Config_Peso_Minimo_Cazoleta AND
Config_Numero_Lineas >= 5) THEN
            Alarma[9]:=TRUE;
            END_IF

// Alarma [10] paso cazoletas en línea 6
 IF (EDGEPOS(gControl.User.di_Trigger) AND Calculos_Linea[6].Peso_Actual < Config_Peso_Minimo_Cazoleta AND
Config_Numero_Lineas >= 6) THEN
            Alarma[10]:=TRUE;
            END_IF
 // Alarma [11] paso cazoletas en línea 7
 IF (EDGEPOS(gControl.User.di_Trigger) AND Calculos_Linea[7].Peso_Actual < Config_Peso_Minimo_Cazoleta AND
Config_Numero_Lineas >= 7) THEN
            Alarma[11]:=TRUE;
            END_IF

// Alarma [12] paso cazoletas en línea 8
 IF (EDGEPOS(gControl.User.di_Trigger) AND Calculos_Linea[8].Peso_Actual < Config_Peso_Minimo_Cazoleta AND
Config_Numero_Lineas >= 8) THEN
            Alarma[12]:=TRUE;
            END_IF

(*------------------------------------------ RESETEAMOS TODAS LAS ALARMAS ------------------------------------------*)
 IF EDGEPOS(Aceptar_Alarmas) THEN
            FOR i:=0 TO 50 DO
               Alarma[i]:=FALSE;
   END_FOR
            END_IF

 IF (Alarma[0]=TRUE OR Alarma[5]=TRUE OR Alarma[6]=TRUE OR Alarma[7]=TRUE OR Alarma[8]=TRUE OR
Alarma[9]=TRUE OR Alarma[10]=TRUE OR Alarma[11]=TRUE OR Alarma[12]=TRUE) AND
Button_Desactivar_Alarma=FALSE THEN
            Rele_Seguridad_Calibradora:= FALSE;
            ELSE
```

```
            Rele_Seguridad_Calibradora:= TRUE;
            END_IF
END_PROGRAM
```

4.3.8. Estadísticas

El módulo de estadísticas lo requerimos para imprimir y guardar todos los datos de de la fruta que ha procesado la calibradora durante un trabajo (nombre del cliente, número de piezas por calibre, número de piezas por salida, etc.)

```
(*********************************************************************
 * COPYRIGHT --
 *********************************************************************
 * Program: Estadisticas
 * File: Estadisticas.st
 * Author: Erico Cruz
 * Created: March 24, 2010
 *********************************************************************
 * Implementation of program Estadisticas
 *********************************************************************)
PROGRAM _INIT
Recipe.Variable[0] := 0;
Recipe.Variable[1] := 1;
END_PROGRAM
PROGRAM _CYCLIC
(*---------------------------------------- INICIO DE ESTADÍSTICAS ----------------------------------------------------------------*)
 IF EDGEPOS(Button_Iniciar_Estadisticas) THEN
            Estadisticas_por_Calibre.Nombre_Variedad:=Variedad_de_la_Fruta_Lectura.Nombre_Variedad;
            CASE Pantalla_Layaut.Modo_de_Calibracion OF
               0:       Estadisticas_por_Calibre.Modo_de_Calibracion:='Peso';
                        1:  Estadisticas_por_Calibre.Modo_de_Calibracion:='Peso y Color';
                        2:  Estadisticas_por_Calibre.Modo_de_Calibracion:='Peso, Color y Diametro';
                        3:  Estadisticas_por_Calibre.Modo_de_Calibracion:='Diametro';
                        4:  Estadisticas_por_Calibre.Modo_de_Calibracion:='Diametro y Color';
                        5:  Estadisticas_por_Calibre.Modo_de_Calibracion:='Color';
                        6:  Estadisticas_por_Calibre.Modo_de_Calibracion:='Peso y Diametro';
            END_CASE
            FOR i:=1 TO 16 DO
            Estadisticas_por_Calibre.Nombre_del_Calibre[i]:=Variedad_de_la_Fruta_Lectura.Nombre_del_Grado[i];
            END_FOR

   FOR i:=0 TO 16 DO
            END_FOR

(* LEER LA FECHA Y HORA DEL PLC  *)
            DTGetTime_0(enable := 1);
            Date_Time := DTGetTime_0.DT1;
            (* The Date and Time type is broken independly *)
```

```
        DT_TO_DTStructure(Date_Time, ADR(Date_Time_Structure));

        (* To each part of the Date and Time is assigned a string *)
        itoa(Date_Time_Structure.day,ADR(strDay));
        itoa(Date_Time_Structure.month,ADR(strMonth));
        itoa(Date_Time_Structure.year,ADR(strYear));
        itoa(Date_Time_Structure.hour,ADR(strHour));
        itoa(Date_Time_Structure.minute,ADR(strMinute));
        itoa(Date_Time_Structure.second,ADR(strSecond));

// Copiamos los datos de fecha y hora en las variables de estadísticas
    // Hora de inicio
        strcpy(ADR(Estadisticas_por_Calibre.Hora_de_Inicio),ADR(''));
        strcat(ADR(Estadisticas_por_Calibre.Hora_de_Inicio),ADR(strHour));
        strcat(ADR(Estadisticas_por_Calibre.Hora_de_Inicio),ADR(':'));
        IF Date_Time_Structure.minute < 10 THEN
          strcat(ADR(Estadisticas_por_Calibre.Hora_de_Inicio),ADR('0'));
          END_IF
        strcat(ADR(Estadisticas_por_Calibre.Hora_de_Inicio),ADR(strMinute));
        strcat(ADR(Estadisticas_por_Calibre.Hora_de_Inicio),ADR(':'));
        IF Date_Time_Structure.second < 10 THEN
          strcat(ADR(Estadisticas_por_Calibre.Hora_de_Inicio),ADR('0'));
          END_IF
        strcat(ADR(Estadisticas_por_Calibre.Hora_de_Inicio),ADR(strSecond));

        // Fecha de inicio
        strcpy(ADR(Estadisticas_por_Calibre.Fecha_de_Inicio),ADR(''));
        IF Date_Time_Structure.day < 10 THEN
          strcat(ADR(Estadisticas_por_Calibre.Fecha_de_Inicio),ADR('0'));
          END_IF
        strcat(ADR(Estadisticas_por_Calibre.Fecha_de_Inicio),ADR(strDay));
        strcat(ADR(Estadisticas_por_Calibre.Fecha_de_Inicio),ADR('/'));
        IF Date_Time_Structure.month < 10 THEN
          strcat(ADR(Estadisticas_por_Calibre.Fecha_de_Inicio),ADR('0'));
          END_IF
        strcat(ADR(Estadisticas_por_Calibre.Fecha_de_Inicio),ADR(strMonth));
        strcat(ADR(Estadisticas_por_Calibre.Fecha_de_Inicio),ADR('/'));
        strcat(ADR(Estadisticas_por_Calibre.Fecha_de_Inicio),ADR(strYear));

        strcpy(ADR(Estadisticas_por_Calibre.Fecha_de_Fin),ADR(''));
        strcpy(ADR(Estadisticas_por_Calibre.Hora_de_Fin),ADR(''));

        DTGetTime_0(enable := 0);

(* BORRAR ESTADÍSTICAS *)
 FOR i:=0 TO 20 DO
          Estadisticas_por_Calibre.Piezas_por_Calibre[i]:=0;
          Estadisticas_por_Calibre.Peso_por_Calibre_Gramos[i]:=0;
          END_FOR
```

```
Estadisticas_por_Calibre.Peso_Total:=0;
Estadisticas_por_Calibre.Piezas_Total:=0;
END_IF

(*---------------------------------------------- FIN DE ESTADÍSTICAS ----------------------------------------------*)
(*--------- con el fin de estadísticas imprimimos todos los dtos a través de una impresora ----------------------------*)
IF EDGENEG(Button_Iniciar_Estadisticas) THEN

(* LEER LA FECHA Y HORA DEL PLC  *)
        DTGetTime_0(enable := 1);
        Date_Time := DTGetTime_0.DT1;
        (* The Date and Time type is broken independly *)
        DT_TO_DTStructure(Date_Time, ADR(Date_Time_Structure));

        (* Para cada una de las partes de la fecha y hora es asignada una palabra STRING *)
        itoa(Date_Time_Structure.day,ADR(strDay));
        itoa(Date_Time_Structure.month,ADR(strMonth));
        itoa(Date_Time_Structure.year,ADR(strYear));
        itoa(Date_Time_Structure.hour,ADR(strHour));
        itoa(Date_Time_Structure.minute,ADR(strMinute));
        itoa(Date_Time_Structure.second,ADR(strSecond));

// Copiamos los datos de fecha y hora en las variables de estadísticas
        strcpy(ADR(Estadisticas_por_Calibre.Hora_de_Fin),ADR(''));
        strcat(ADR(Estadisticas_por_Calibre.Hora_de_Fin),ADR(strHour));
        strcat(ADR(Estadisticas_por_Calibre.Hora_de_Fin),ADR(':'));
        IF Date_Time_Structure.minute < 10 THEN
          strcat(ADR(Estadisticas_por_Calibre.Hora_de_Fin),ADR('0'));
          END_IF
        strcat(ADR(Estadisticas_por_Calibre.Hora_de_Fin),ADR(strMinute));
        strcat(ADR(Estadisticas_por_Calibre.Hora_de_Fin),ADR(':'));
        IF Date_Time_Structure.second < 10 THEN
          strcat(ADR(Estadisticas_por_Calibre.Hora_de_Fin),ADR('0'));
          END_IF
        strcat(ADR(Estadisticas_por_Calibre.Hora_de_Fin),ADR(strSecond));

        // Fecha de fin
        strcpy(ADR(Estadisticas_por_Calibre.Fecha_de_Fin),ADR(''));
        IF Date_Time_Structure.day < 10 THEN
          strcat(ADR(Estadisticas_por_Calibre.Fecha_de_Fin),ADR('0'));
          END_IF
        strcat(ADR(Estadisticas_por_Calibre.Fecha_de_Fin),ADR(strDay));
        strcat(ADR(Estadisticas_por_Calibre.Fecha_de_Fin),ADR('/'));
        IF Date_Time_Structure.month < 10 THEN
          strcat(ADR(Estadisticas_por_Calibre.Fecha_de_Fin),ADR('0'));
          END_IF
        strcat(ADR(Estadisticas_por_Calibre.Fecha_de_Fin),ADR(strMonth));
        strcat(ADR(Estadisticas_por_Calibre.Fecha_de_Fin),ADR('/'));
        strcat(ADR(Estadisticas_por_Calibre.Fecha_de_Fin),ADR(strYear));
```

```
            DTGetTime_0(enable := 0);
            Recipe:=Estadisticas_por_Calibre;
            Guardar_Estadisticas:=TRUE;

 END_IF

(* ------------------------------------ GUARDAMOS LAS ESTADÍSTICAS EN MEMÓRIA ----------------------------*)
IF Guardar_Estadisticas=TRUE THEN
 FOR a:=0 TO 99 DO
            IF (Nombre_Fichero[a]='ero') OR (Nombre_Fichero[a]='Nuevo fichero') THEN
              x:=a;
            END_IF
 END_FOR
 END_IF

// Enviamos nombre y numero de fichero
 IF EDGEPOS(Guardar_Estadisticas) THEN
            Step_1:=TRUE;
            Step_2:=FALSE;
   Step_3:= FALSE;
            END_IF

 IF Step_1=TRUE THEN
            Indice_Fichero:=x;
            strcpy(ADR(NuevoNombre),ADR(''));

            IF Date_Time_Structure.day < 10 THEN
                        strcat(ADR(NuevoNombre),ADR('0'));
                        strcat(ADR(NuevoNombre),ADR(strDay));
                        ELSE
                        strcat(ADR(NuevoNombre),ADR(strDay));
                        END_IF
            strcat(ADR(NuevoNombre),ADR('_'));

            IF Date_Time_Structure.month < 10 THEN
                        strcat(ADR(NuevoNombre),ADR('0'));
                        strcat(ADR(NuevoNombre),ADR(strMonth));
                        ELSE
                        strcat(ADR(NuevoNombre),ADR(strMonth));
                        END_IF

            strcat(ADR(NuevoNombre),ADR('_'));

            IF strYear='2010' THEN
                        strcat(ADR(NuevoNombre),ADR('10'));
            ELSIF strYear='2011' THEN
                        strcat(ADR(NuevoNombre),ADR('11'));
            ELSIF strYear='2012' THEN
```

```
            strcat(ADR(NuevoNombre),ADR('12'));
ELSIF strYear='2013' THEN
            strcat(ADR(NuevoNombre),ADR('13'));
ELSIF strYear='2014' THEN
            strcat(ADR(NuevoNombre),ADR('14'));
ELSIF strYear='2015' THEN
            strcat(ADR(NuevoNombre),ADR('15'));
ELSIF strYear='2016' THEN
            strcat(ADR(NuevoNombre),ADR('16'));
ELSIF strYear='2017' THEN
            strcat(ADR(NuevoNombre),ADR('17'));
ELSIF strYear='2018' THEN
            strcat(ADR(NuevoNombre),ADR('18'));
ELSIF strYear='2019' THEN
            strcat(ADR(NuevoNombre),ADR('19'));
ELSIF strYear='2020' THEN
            strcat(ADR(NuevoNombre),ADR('20'));
END_IF

strcat(ADR(NuevoNombre),ADR('_'));

IF Date_Time_Structure.hour < 10 THEN
            strcat(ADR(NuevoNombre),ADR('0'));
            strcat(ADR(NuevoNombre),ADR(strHour));
            ELSE
            strcat(ADR(NuevoNombre),ADR(strHour));
            END_IF

strcat(ADR(NuevoNombre),ADR('h'));

IF Date_Time_Structure.minute < 10 THEN
            strcat(ADR(NuevoNombre),ADR('0'));
            strcat(ADR(NuevoNombre),ADR(strMinute));
            ELSE
            strcat(ADR(NuevoNombre),ADR(strMinute));
            END_IF

strcat(ADR(NuevoNombre),ADR('m'));

IF Date_Time_Structure.second < 10 THEN
            strcat(ADR(NuevoNombre),ADR('0'));
            strcat(ADR(NuevoNombre),ADR(strSecond));
            ELSE
            strcat(ADR(NuevoNombre),ADR(strSecond));
            END_IF

strcat(ADR(NuevoNombre),ADR('s'));
strcat(ADR(NuevoNombre),ADR('_'));
strcat(ADR(NuevoNombre),ADR(Estadisticas_por_Calibre.Nombre_Agricultor));
```

```
            END_IF
// Enviamos señal de guardar
 IF (Step_1=TRUE AND CSV_Signal_1=TRUE) THEN
            Step_2:=TRUE;
   Step_1:=FALSE;
            Step_3:=FALSE;
            END_IF

 IF Step_2=TRUE THEN
            CMDEscribir:=TRUE;
   END_IF

 // Reseteamos bits
 IF (Step_2=TRUE AND CSV_Signal_2=TRUE) THEN
            Step_2:=FALSE;
   Step_1:=FALSE;
            Step_3:=TRUE;
            END_IF

 IF Step_3=TRUE THEN
            CMDEscribir:=FALSE;
            Guardar_Estadisticas:=FALSE;
   Step_2:=FALSE;
   Step_1:=FALSE;
            Step_3:=FALSE;
            END_IF
(*---------------------------------------- RECUPERAMOS ESTADÍSTICAS DE LA BASE DE DATOS ------------------------------*)
 IF EDGEPOS(Leer_Recipe) THEN
   Esperar_Datos:=TRUE;
   END_IF

 TON_0(IN:=Esperar_Datos, PT := T#4s);

 IF (Esperar_Datos=TRUE AND EDGEPOS(TON_0.Q)) THEN
  Estadisticas_por_Calibre:=Recipe;
  Esperar_Datos:=FALSE;
  END_IF

END_PROGRAM
```

4.3.9. Cálculos estadísticas

Con el módulo de cálculos de estadísticas calculamos los valores que posteriormente guardamos en el bloque “Estadísticas” antes descrito.

```
(*********************************************************************
 * COPYRIGHT --
 *********************************************************************
 * Program: Calculos_Estadisticas
 * File: Calculos_Estadisticas.st
 * Author: Erico Cruz
 * Created: April 17, 2010
 *********************************************************************
 * Implementation of program Calculos_Estadisticas
 *********************************************************************)
PROGRAM _INIT
(* TODO : Add your code here *)
END_PROGRAM

PROGRAM _CYCLIC
(*------------------------------ CALCULAMOS TODOS LOS VALORES DE LAS ESTADÍSTICAS ----------------------------*)
Estadisticas_por_Calibre.Piezas_Total:=Estadisticas_por_Calibre.Piezas_por_Calibre[1] +
Estadisticas_por_Calibre.Piezas_por_Calibre[2] + Estadisticas_por_Calibre.Piezas_por_Calibre[3] +
Estadisticas_por_Calibre.Piezas_por_Calibre[4] + Estadisticas_por_Calibre.Piezas_por_Calibre[5] +
Estadisticas_por_Calibre.Piezas_por_Calibre[6] + Estadisticas_por_Calibre.Piezas_por_Calibre[7] +
Estadisticas_por_Calibre.Piezas_por_Calibre[8] + Estadisticas_por_Calibre.Piezas_por_Calibre[9] +
Estadisticas_por_Calibre.Piezas_por_Calibre[10] + Estadisticas_por_Calibre.Piezas_por_Calibre[11] +
Estadisticas_por_Calibre.Piezas_por_Calibre[12] + Estadisticas_por_Calibre.Piezas_por_Calibre[13] +
Estadisticas_por_Calibre.Piezas_por_Calibre[14] + Estadisticas_por_Calibre.Piezas_por_Calibre[15] +
Estadisticas_por_Calibre.Piezas_por_Calibre[16];
  Estadisticas_por_Calibre.Peso_Total:=Estadisticas_por_Calibre.Peso_por_Calibre[1] +
Estadisticas_por_Calibre.Peso_por_Calibre[2] + Estadisticas_por_Calibre.Peso_por_Calibre[3] +
Estadisticas_por_Calibre.Peso_por_Calibre[4] + Estadisticas_por_Calibre.Peso_por_Calibre[5] +
Estadisticas_por_Calibre.Peso_por_Calibre[6] + Estadisticas_por_Calibre.Peso_por_Calibre[7] +
Estadisticas_por_Calibre.Peso_por_Calibre[8] + Estadisticas_por_Calibre.Peso_por_Calibre[9] +
Estadisticas_por_Calibre.Peso_por_Calibre[10] + Estadisticas_por_Calibre.Peso_por_Calibre[11] +
Estadisticas_por_Calibre.Peso_por_Calibre[12] + Estadisticas_por_Calibre.Peso_por_Calibre[13] +
Estadisticas_por_Calibre.Peso_por_Calibre[14] + Estadisticas_por_Calibre.Peso_por_Calibre[15] +
Estadisticas_por_Calibre.Peso_por_Calibre[16];

  Peso_Total:=Estadisticas_por_Calibre.Peso_Total;
  Piezas_Total:=Estadisticas_por_Calibre.Piezas_Total;

 FOR i:=1 TO 16 DO
            IF Peso_Total > 0 THEN
       Estadisticas_por_Calibre.Porcentaje_Peso_por_Calibre[i]:=100*Estadisticas_por_Calibre.Peso_por_Calibre[i] /
Peso_Total;
              ELSE
              Estadisticas_por_Calibre.Porcentaje_Peso_por_Calibre[i]:=0;
            END_IF

            IF Piezas_Total > 0 THEN
```

```
Estadisticas_por_Calibre.Porcentaje_Piezas_por_Calibre[i]:=100*Estadisticas_por_Calibre.Piezas_por_Calibre[i] /
Piezas_Total;
            ELSE
              Estadisticas_por_Calibre.Porcentaje_Piezas_por_Calibre[i]:=0;
          END_IF
          Estadisticas_por_Calibre.Peso_por_Calibre[i]:=Estadisticas_por_Calibre.Peso_por_Calibre_Gramos[i]/1000;
 END_FOR
END_PROGRAM
```

4.3.10. Bloque Server

El bloque de programa server contiene el código que abre el puerto de comunicación TCP server, recibe los datos de la CPU de las cámaras y los convierte a variables del tipo real para que posteriormente las gestionamos en el bloque “Cálculo Cámaras” antes descrito.

```
(*********************************************************************
 * COPYRIGHT -- Bernecker + Rainer
 *********************************************************************
 * Program: Server
 * File: Server.st
 * Author: Erico Cruz
 * Created: February 21, 2008
 *********************************************************************
 * Implementation of program Server
 *********************************************************************)

PROGRAM _CYCLIC
        CASE Server.sStep OF
                0:  (* PASO 0: ABRIMOS LA INTERFAZ DE ETHERNET *)
                        Server.TcpOpen_0.enable := 1;
                        Server.TcpOpen_0.pIfAddr := 0;  (* Listen on all TCP/IP Interfaces*)
                        Server.TcpOpen_0.port := 9600;  (* Port to listen*)
                        Server.TcpOpen_0.options := 0;
                        Server.TcpOpen_0;  (* Call the Function*)

                        IF Server.TcpOpen_0.status = 0 THEN  (* TcpOpen successfull*)
                                Server.sStep := 5;
                        ELSIF Server.TcpOpen_0.status = ERR_FUB_BUSY THEN  (* TcpOpen not finished ->
redo *)
                        ELSE  (* VAMOS AL PASO DE ERROR *)
                                Server.sStep := 100;
                        END_IF

                5:
                        Server.linger_opt.lLinger := 0; (* linger Time = 0 *)
```

```
                Server.linger_opt.lOnOff := 1;
                Server.TcpIoctl_0.enable := 1;
                Server.TcpIoctl_0.ident := Server.TcpOpen_0.ident;
                Server.TcpIoctl_0.ioctl := tcpSO_LINGER_SET;
                Server.TcpIoctl_0.pData := ADR(Server.linger_opt);
                Server.TcpIoctl_0.datalen := SIZEOF(Server.linger_opt);
                Server.TcpIoctl_0;

                IF Server.TcpIoctl_0.status = 0 THEN  (*TCP FINALIZADO *)
                        Server.sStep := 10;

                ELSIF Server.TcpIoctl_0.status = ERR_FUB_BUSY THEN  (* TCP NO FINALIZADO *)

                 (* OCUPADO*)
                ELSE  (* VAMOS A PASO ERROR *)
                        Server.sStep := 100;
                END_IF

        10: (*ESPERAMOS PARA CONEXIÓN CLIENTE *)
                Server.TcpServer_0.enable := 1;
                Server.TcpServer_0.ident := Server.TcpOpen_0.ident;  (*CONEXIÓN IDENT DE
AsTCP.TCP_Open *)
                Server.TcpServer_0.backlog := 1;  (* VARIOS CLIENTES ESPERANDO CONEXIÓN
SIMULTANEA*)
                Server.TcpServer_0.pIpAddr := ADR(Server.client_address);  (* DONDE ESCRIBIR LA
DIRECCIÓN IP DEL CLIENTE*)
                Server.TcpServer_0;  (* LLAMAR FUNCIÓN*)

                IF Server.TcpServer_0.status = 0 THEN  (* Status = 0 SI UN CLIENTE CONECTADO A
SERVIDOR *)
                        Server.sStep := 15;
                ELSIF Server.TcpServer_0.status = ERR_FUB_BUSY THEN  (* TCP SERVER NO
FINALIZADO *)
                 (* OCUPADO *)
                ELSE  (* VAMOS A PASO ERROR, PASO 100 *)
                        Server.sStep := 100;
                END_IF

        15:
                Server.TcpIoctl_0.enable := 1;
                Server.TcpIoctl_0.ident := Server.TcpServer_0.identclnt;
                Server.TcpIoctl_0.ioctl := tcpSO_LINGER_SET;
                Server.TcpIoctl_0.pData := ADR(Server.linger_opt);
                Server.TcpIoctl_0.datalen := SIZEOF(Server.linger_opt);
                Server.TcpIoctl_0;

                IF Server.TcpIoctl_0.status = 0 THEN  (* TCP FINALIZADO *)
                        Server.sStep := 20;
```

```
                    ELSIF Server.TcpIoctl_0.status = ERR_FUB_BUSY THEN  (* TCP NO FINALIZADO*)

                     (* OCUPADO *)
                    ELSE  (* VAMOS A PASO ERROR *)
                        Server.sStep := 100;
                    END_IF

                20: (* ESPERAMOS DATOS *)
                    Server.TcpRecv_0.enable := 1;
                    Server.TcpRecv_0.ident := Server.TcpServer_0.identclnt;
                    Server.TcpRecv_0.pData      := ADR(Server.data_buffer);  (* DONDE ALMACENAMOS
LOS DATOS RECIBIDOS *)
                    Server.TcpRecv_0.datamax := SIZEOF(Server.data_buffer);  (* LONGITUD DEL BUFFER
DE LOS DATOS RECIBIDOS *)
                    Server.TcpRecv_0.flags := 0;
                    Server.TcpRecv_0;  (* LLAMAR LA FUNCIÓN*)

                    IF Server.TcpRecv_0.status = 0 THEN  (* DATOS RECIBIDOS *)
                        Server.sStep :=20;
                        counter := counter + 1;
                        //Server.sStep := 30;
                        Server.recv_timeout := 0;

                        IF counter > 8 THEN
            (*---------------- CONVERTIMOS TODOS LOS DATOS DE STRING RECIBIDOS A REAL----------------------*)
            Camaras_Grado_REAL.Linea[1].Posicion_1.Diametro_Avg :=atof(ADR(Server.data_buffer[0]));
            Camaras_Grado_REAL.Linea[1].Posicion_1.Diametro_Max :=atof(ADR(Server.data_buffer[1]));
            Camaras_Grado_REAL.Linea[1].Posicion_1.Diametro_Min :=atof(ADR(Server.data_buffer[2]));
            Camaras_Grado_REAL.Linea[1].Posicion_2.Diametro_Avg :=atof(ADR(Server.data_buffer[3]));
            Camaras_Grado_REAL.Linea[1].Posicion_2.Diametro_Max :=atof(ADR(Server.data_buffer[4]));
            Camaras_Grado_REAL.Linea[1].Posicion_2.Diametro_Min :=atof(ADR(Server.data_buffer[5]));
            Camaras_Grado_REAL.Linea[1].Posicion_3.Diametro_Avg :=atof(ADR(Server.data_buffer[6]));
            Camaras_Grado_REAL.Linea[1].Posicion_3.Diametro_Max :=atof(ADR(Server.data_buffer[7]));
            Camaras_Grado_REAL.Linea[1].Posicion_3.Diametro_Min :=atof(ADR(Server.data_buffer[8]));
            Camaras_Grado_REAL.Linea[1].Posicion_4.Diametro_Avg :=atof(ADR(Server.data_buffer[9]));
            Camaras_Grado_REAL.Linea[1].Posicion_4.Diametro_Max :=atof(ADR(Server.data_buffer[10]));
            Camaras_Grado_REAL.Linea[1].Posicion_4.Diametro_Min :=atof(ADR(Server.data_buffer[11]));
            Camaras_Grado_REAL.Linea[2].Posicion_1.Diametro_Avg :=atof(ADR(Server.data_buffer[12]));
            Camaras_Grado_REAL.Linea[2].Posicion_1.Diametro_Max :=atof(ADR(Server.data_buffer[13]));
            Camaras_Grado_REAL.Linea[2].Posicion_1.Diametro_Min :=atof(ADR(Server.data_buffer[14]));
            Camaras_Grado_REAL.Linea[2].Posicion_2.Diametro_Avg :=atof(ADR(Server.data_buffer[15]));
            Camaras_Grado_REAL.Linea[2].Posicion_2.Diametro_Max :=atof(ADR(Server.data_buffer[16]));
            Camaras_Grado_REAL.Linea[2].Posicion_2.Diametro_Min :=atof(ADR(Server.data_buffer[17]));
            Camaras_Grado_REAL.Linea[2].Posicion_3.Diametro_Avg :=atof(ADR(Server.data_buffer[18]));
            Camaras_Grado_REAL.Linea[2].Posicion_3.Diametro_Max :=atof(ADR(Server.data_buffer[19]));
            Camaras_Grado_REAL.Linea[2].Posicion_3.Diametro_Min :=atof(ADR(Server.data_buffer[20]));
            Camaras_Grado_REAL.Linea[2].Posicion_4.Diametro_Avg :=atof(ADR(Server.data_buffer[21]));
            Camaras_Grado_REAL.Linea[2].Posicion_4.Diametro_Max :=atof(ADR(Server.data_buffer[22]));
            Camaras_Grado_REAL.Linea[2].Posicion_4.Diametro_Min :=atof(ADR(Server.data_buffer[23]));
```

```
            Camaras_Grado_REAL.Linea[3].Posicion_1.Diametro_Avg :=atof(ADR(Server.data_buffer[24]));
            Camaras_Grado_REAL.Linea[3].Posicion_1.Diametro_Max :=atof(ADR(Server.data_buffer[25]));
            Camaras_Grado_REAL.Linea[3].Posicion_1.Diametro_Min :=atof(ADR(Server.data_buffer[26]));
            Camaras_Grado_REAL.Linea[3].Posicion_2.Diametro_Avg :=atof(ADR(Server.data_buffer[27]));
            Camaras_Grado_REAL.Linea[3].Posicion_2.Diametro_Max :=atof(ADR(Server.data_buffer[28]));
            Camaras_Grado_REAL.Linea[3].Posicion_2.Diametro_Min :=atof(ADR(Server.data_buffer[29]));
            Camaras_Grado_REAL.Linea[3].Posicion_3.Diametro_Avg :=atof(ADR(Server.data_buffer[30]));
            Camaras_Grado_REAL.Linea[3].Posicion_3.Diametro_Max :=atof(ADR(Server.data_buffer[31]));
            Camaras_Grado_REAL.Linea[3].Posicion_3.Diametro_Min :=atof(ADR(Server.data_buffer[32]));
            Camaras_Grado_REAL.Linea[3].Posicion_4.Diametro_Avg :=atof(ADR(Server.data_buffer[33]));
            Camaras_Grado_REAL.Linea[3].Posicion_4.Diametro_Max :=atof(ADR(Server.data_buffer[34]));
            Camaras_Grado_REAL.Linea[3].Posicion_4.Diametro_Min :=atof(ADR(Server.data_buffer[35]));
            Camaras_Grado_REAL.Linea[4].Posicion_1.Diametro_Avg :=atof(ADR(Server.data_buffer[36]));
            Camaras_Grado_REAL.Linea[4].Posicion_1.Diametro_Max :=atof(ADR(Server.data_buffer[37]));
            Camaras_Grado_REAL.Linea[4].Posicion_1.Diametro_Min :=atof(ADR(Server.data_buffer[38]));
            Camaras_Grado_REAL.Linea[4].Posicion_2.Diametro_Avg :=atof(ADR(Server.data_buffer[39]));
            Camaras_Grado_REAL.Linea[4].Posicion_2.Diametro_Max :=atof(ADR(Server.data_buffer[40]));
            Camaras_Grado_REAL.Linea[4].Posicion_2.Diametro_Min :=atof(ADR(Server.data_buffer[41]));
            Camaras_Grado_REAL.Linea[4].Posicion_3.Diametro_Avg :=atof(ADR(Server.data_buffer[42]));
            Camaras_Grado_REAL.Linea[4].Posicion_3.Diametro_Max :=atof(ADR(Server.data_buffer[43]));
            Camaras_Grado_REAL.Linea[4].Posicion_3.Diametro_Min :=atof(ADR(Server.data_buffer[44]));
            Camaras_Grado_REAL.Linea[4].Posicion_4.Diametro_Avg :=atof(ADR(Server.data_buffer[45]));
            Camaras_Grado_REAL.Linea[4].Posicion_4.Diametro_Max :=atof(ADR(Server.data_buffer[46]));
            Camaras_Grado_REAL.Linea[4].Posicion_4.Diametro_Min :=atof(ADR(Server.data_buffer[47]));
            i:=1;
Camaras_Grado.Linea[i].Posicion_1.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_1.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_1.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_1.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_1.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_1.Diametro_Min);
Camaras_Grado.Linea[i].Posicion_2.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_2.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_2.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_2.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_2.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_2.Diametro_Min);
Camaras_Grado.Linea[i].Posicion_3.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_3.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_3.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_3.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_3.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_3.Diametro_Min);
Camaras_Grado.Linea[i].Posicion_4.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_4.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_4.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_4.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_4.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_4.Diametro_Min);
```

```
i:=2;
Camaras_Grado.Linea[i].Posicion_1.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_1.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_1.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_1.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_1.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_1.Diametro_Min);
Camaras_Grado.Linea[i].Posicion_2.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_2.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_2.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_2.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_2.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_2.Diametro_Min);
Camaras_Grado.Linea[i].Posicion_3.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_3.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_3.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_3.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_3.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_3.Diametro_Min);
Camaras_Grado.Linea[i].Posicion_4.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_4.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_4.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_4.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_4.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_4.Diametro_Min);

i:=3;
Camaras_Grado.Linea[i].Posicion_1.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_1.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_1.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_1.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_1.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_1.Diametro_Min);
Camaras_Grado.Linea[i].Posicion_2.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_2.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_2.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_2.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_2.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_2.Diametro_Min);
Camaras_Grado.Linea[i].Posicion_3.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_3.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_3.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_3.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_3.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_3.Diametro_Min);
Camaras_Grado.Linea[i].Posicion_4.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_4.Diametro_Avg);
```

```
Camaras_Grado.Linea[i].Posicion_4.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_4.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_4.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_4.Diametro_Min);

i:=4;
Camaras_Grado.Linea[i].Posicion_1.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_1.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_1.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_1.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_1.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_1.Diametro_Min);
Camaras_Grado.Linea[i].Posicion_2.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_2.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_2.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_2.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_2.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_2.Diametro_Min);
Camaras_Grado.Linea[i].Posicion_3.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_3.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_3.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_3.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_3.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_3.Diametro_Min);
Camaras_Grado.Linea[i].Posicion_4.Diametro_Avg :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_4.Diametro_Avg);
Camaras_Grado.Linea[i].Posicion_4.Diametro_Max :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_4.Diametro_Max);
Camaras_Grado.Linea[i].Posicion_4.Diametro_Min :=
REAL_TO_INT(Camaras_Grado_REAL.Linea[i].Posicion_4.Diametro_Min);

counter := 0;
Trigger_Camaras := TRUE;
END_IF
                    100:  (* EN ESTE PASO IMPLEMENTAMOS LO QUE QUEREMOS EN CASO DE ERROR DE
COMUNICACIÓN *)
Server.sStep := 100; (*INICIALIZAMOS OTRA VEZ LA COMUNICACIÓN, PASO 0*)
          END_CASE
END_PROGRAM
```

5. PRESUPUESTO

5.1. Presupuesto mecánico

Para la instalación del equipo electrónico en la calibradora requerimos de una caja mecánica donde ubicar las cámaras y otra donde ubicar la CPU con pantalla táctil.

5.1.1. Conjunto caja cámaras

Descripción	Proveedor	Unidades	P. Unitario	P. Total
CONJUNTO MECÁNICO CAJA DE CAMARAS				
Perfil Bosch 45x45L R L=850mm	Alutecman	4,00	20,27	81,08
Perfil Bosch 45x45L R L=610mm	Alutecman	2,00	16,51	33,02
Perfil Bosch 45x45L R L=1200mm	Alutecman	4,00	25,91	103,64
Perfil Bosch 30x30 L=610mm	Alutecman	4,00	9,69	38,76
Perfil Bosch 30x30 L=865mm	Alutecman	1,00	11,76	11,76
Perfil Bosch 30x30 L=850mm	Alutecman	2,00	11,76	23,52
Perfil Bosch 30x30 L=625mm	Alutecman	3,00	9,69	29,07
Rinconera esférica Bosch 45x45	Alutecman	4,00	6,40	25,60
Tapa rinconera esférica Bosch 45x45	Alutecman	4,00	1,78	7,12
Esquadra Bosch Variofix S40/45 M8x25 R10	Alutecman	20,00	5,18	103,60
Bosch Variofix-block S40/45	Alutecman	8,00	2,37	18,96
Esquadra Bosch 30x30 M6x14 R8	Alutecman	16,00	2,97	47,52
Cabeza de martillo Bosch M6x25 R8	Alutecman	9,00	0,50	4,50
Tuerca para cab. martillo Bosch M6	Alutecman	9,00	0,13	1,17
Tuerca de martillo M8 R10	Alutecman	8,00	0,44	3,52
Tornillo din 912 M8x16	Ferretería Puig	8,00	0,04	0,28
Arandela din 125 M8	Ferretería Puig	8,00	0,02	0,16
Arandela DIN 127B M3	Ferretería Puig	8,00	0,02	0,16
Tuerca DIN 934 M4	Ferretería Puig	12,00	0,04	0,48
Tornillo DIN 912 M4x8	Ferretería Puig	8,00	0,06	0,46
Tornillo DIN 912 M4x25	Ferretería Puig	12,00	0,03	0,30
Platina unión perfiles armario	ELPA	4,00	5,00	20,00
Soporte cámaras Infaimon	ELPA	2,00	8,00	16,00
Arandela de presión din-127B M4	Ferretería Puig	42,00	0,02	0,84
Tornillo Allen din 912 M4x10	Ferretería Puig	18,00	0,04	0,63
Soporte Iluminación Infaimon	ELPA	3,00	12,00	36,00
Carenado 1 caja de cámaras	ELPA	2,00	47,00	94,00
Carenado 2 caja de cámaras	ELPA	2,00	43,50	87,00
Carenado 3 caja de cámaras	ELPA	1,00	38,50	38,50
Tornillo ISO 7380 M5x25	Ferretería Puig	20,00	0,27	5,39
			PRECIO TOTAL:	**833,05 €**

Tabla 2

5.1.2. Conjunto caja de CPU

Descripción	Proveedor	Unidades	P. Unitario	P. Total
CONJUNTO MECÁNICO CAJA PANTALLA TÁCTIL + CPU				
Caja pantalla B&R PP420 10,4"	ELPA	1,00	175,00	175,00
Tapa caja pantalla B&R PP420 10,4"	ELPA	1,00	15,00	15,00
Asa 160B20 Aluminio anodizado	EMKA	1,00	8,43	8,43
Caja CPU	ELPA	1,00	200,00	200,00
Tapa caja CPU	ELPA	1,00	20,00	20,00
Racor eléctrico M20x1,5	E. Importados	2,00	0,70	1,40
Racor eléctrico M25x1,5	E. Importados	1,00	0,70	0,70
Tornillo ISO 7380 M4x8	Ferretería Puig	8,00	0,16	1,32
Arandela Din 125 M6	Ferretería Puig	2,00	0,02	0,04
Tornillo Din 912 M6x10	Ferretería Puig	2,00	0,06	0,11
Ventilador extractor 24Vdc	E. Importados	2,00	12,00	24,00
			PRECIO TOTAL:	**446,00 €**

Tabla 3

5.2. Presupuesto eléctrico electrónico

Para la implementación de la parte electrónica trabajaremos con tres proveedores:

- JasVision: Material de visión artificial
- Cuadro eléctrico cámaras: ACE Aparellajes y Cuadros Eléctricos
- Bernecker & Reiner B&R: Electrónica PLC
- Cuadro eléctrico módulos PLC: ACE Aparellajes y Cuadros Eléctricos

5.2.1. Material visión artificial

Descripción	Proveedor	Unidades	P. Unitario	P. Total
MATERIAL VISIÓN ARTIFICIAL				
Sherlock 7.x Prof. Lic. USB	JasVision	1,00	2.534,49	2.534,49
Camara M1024-1/3"Monocromo	JasVision	2,00	1.035,63	2.071,26
Barra de LEDs IR400x25mm	JasVision	4,00	365,20	1.460,80
Mounting Bracket	JasVision	2,00	44,63	89,26
Optica alta resolucion 8mm	JasVision	2,00	222,16	444,32
Cable Gige 10M RJ45	JasVision	2,00	46,22	92,44
Conector Hirose 12hembra+cable	JasVision	2,00	45,90	91,80
Fuente alimentacion 12V Genie	JasVision	1,00	43,28	43,28
Filtro IBP850 para M27x0.5	JasVision	2,00	67,00	134,00
Strobe para iluminacion LED	JasVision	1,00	782,00	782,00
Conector CCS 12V + cable	JasVision	4,00	17,00	68,00
Fuente alimentacion 12V	JasVision	1,00	97,75	97,75
			PRECIO TOTAL:	**7.909,40 €**

Tabla 4

5.2.2. Cuadro eléctrico visión artificial

Descripción	Proveedor	Unidades	P. Unitario	P. Total
CUADRO ELÉCTRICO VISIÓN ARTIFICIAL				
Magnetotérmico	ACE	1,00	67,00	67,00
Bornes de contacto	ACE	20,00	0,70	14,00
Disyuntor	ACE	1,00	24,72	24,72
Ventilador extractor	ACE	1,00	18,00	18,00
Cuadro eléctrico	ACE	1,00	180,00	180,00
h. técnico eléctrico	ACE	15,00	18,00	270,00
			PRECIO TOTAL:	**573,72 €**

5.2.3. Electrónica PLC

Descripción	Proveedor	Unidades	P. Unitario	P. Total
ELECTRÓNICA PLC				
POWER PANEL PP420 10" TFT	B&R	1,00	1.457,47	1.457,47
POWERLINK V1/V2 interface	B&R	1,00	67,20	67,20
X20 supply moduler bus control (X20PS9400)	B&R	1,00	29,40	29,40
X20 bus base (X20BB80)	B&R	1,00	15,30	15,30
X20 terminal block 12-pin 24V (X20TB12)	B&R	1,00	4,25	4,25
X20, 4 digital inputs 24Vdc (X20DI4371)	B&R	1,00	17,68	17,68
X20,1 analog input,full-bridge (X20AI1744)	B&R	3,00	133,28	399,84
X20 Analog output module (X20AO4622)	B&R	2,00	122,40	244,80
X20 8 Digital Out, 24VDC,2.0A (X20DO8332)	B&R	10,00	65,96	659,60
X20 supply module for I/O (X20PS2100)	B&R	5,00	14,28	71,40
X20 supply bus module (X20BM01)	B&R	5,00	6,80	34,00
X20 supply bus module 24V (X20BM11)	B&R	16,00	6,80	108,80
X20 terminal block 12-pin 24V (X20TB12)	B&R	21,00	4,25	89,25
			PRECIO TOTAL:	**3.198,99 €**

Tabla 5

5.2.4. Cuadro eléctrico PLC

Descripción	Proveedor	Unidades	P. Unitario	P. Total
CUADRO ELÉCTRICO MÓDULOS ENTRADAS Y SALIDAS PLC				
Seccionador	ACE	1,00	13,15	13,15
Pulsador luminoso verde	ACE	1,00	20,25	20,25
Pulsador de emergencia	ACE	1,00	12,05	12,05
Indicador luminoso blanco	ACE	1,00	20,25	20,25
Indicador luminoso amarillo	ACE	1,00	20,25	20,25
Indicador luminoso rojo	ACE	1,00	20,25	20,25
Fuente de alimentación 24Vdc	ACE	1,00	170,50	170,50
Pulsador negro	ACE	1,00	8,90	8,90
Sirena	ACE	1,00	65,96	65,96
Bornes de 2,5mm	ACE	80,00	0,70	56,00
Cuadro eléctrico	ACE	1,00	320,00	320,00
m cable 1x0,75	ACE	25,00	0,95	23,75
m cable 1x1,5	ACE	25,00	1,15	28,75
Ventilador extractor	ACE	16,00	6,80	108,80
Horas técnico eléctrico	ACE	25,00	4,25	106,25
			PRECIO TOTAL:	**995,11 €**

5.3. Presupuesto TOTAL

Descripción	Proveedor	Unidades	P. Unitario	P. Total
PRESUPUESTO TOTAL				
Conjunto caja de cámaras	ELPA	1,00	833,05	833,05
Conjunto caja de CPU	ELPA	1,00	446,00	446,00
Material visión artificial	JasVision	1,00	7.909,40	7.909,40
Cuadro eléctrico visión artificial	ACE	1,00	573,72	573,72
Electrónica PLC	B&R	1,00	3.198,99	3.198,99
Cuadro eléctrico módulos I/O PLC	ACE	1,00	995,11	995,11
			PRECIO TOTAL:	**13.956,27 €**

Como podemos apreciar, el costo total de la implementación del sistema de visión artificial y el control automático de la calibradora (visión artificial y CPU principal) asciende a 14.000€. Conociendo el mercado, sabemos que un sistema que calibración de la fruta solo por peso tiene un valor medio de 25.000€, con lo que estimo que el presente proyecto podríamos estar ofreciéndolo por 35.000€ disponiendo, por lo tanto, de un margen de contribución de 21.000€, es decir, con este proyecto estamos consiguiendo una rentabilidad del 250%.

6. CONCLUSIONES

Con todo lo explicado en el presente documento, se pueden considerar definidas las características y condiciones que reunirá el módulo de visión artificial en la parte eléctrica, mecánica y automática para la consecución del proyecto. Es importante tener muy presente que con el proyecto desarrollado podemos superar sin el menor inconveniente los objetivos en producción, productividad, y el estándar de calidad reduciendo considerablemente los costos en la mano de obra. Cabe destacar que además de la reducción de la mano de obra y costos fijos también conseguimos un trato mejorado con la fruta resultando en una mayor calidad de esta en el producto final embasado y calibrado.

Así pues, gracias a este proyecto podemos obtener mayores velocidades de calibración, con menores costos de producción, mayor fiabilidad en los resultados de calibración y un mejor trato con el producto ya que no es necesario el empleo de la mano de obra para la consecución de la debida calibración de la fruta, verdura y hortalizas.

6.1. Futuros proyectos

Una buena idea para la consecución de futuros proyectos es la de desarrollar sistemas de visión artificial que detecten el daño en la fruta causado por golpes durante la cadena de procesamiento de esta, que al ojo humano no es detectable hasta que esta ya se

encuentra al final de la cadena, es decir, cuando esta ya se encuentra en la fase de distribución final (los puntos de distribución, tiendas, accesibles al consumidor final). Con el empleo de esta tecnología se podrían evitar grandes pérdidas económicas y una mejora altamente significativa en la calidad del producto final accesible al consumidor.

Otro proyecto considerado altamente útil podría ser la de desarrollar sistemas de visión artificial que detecten el nivel de azúcar en la fruta de modo que nos permita que esta sea seleccionada y/o clasificada según su contenido de azúcar.

7. BIBLIOGRAFÍA

7.1. Calibración y envases de fruta

- Hortyfruta; Junta de Andalucía (2008). Tipificación, Requisitos mínimos de calidad de frutas y hortalizas
  Disponible en:
  http://hortyfruta.es/Images/folletotipificacion.pdf

- InfoAgro Systems, S.L. (2010). Calibre universal para fruta
  http://www.paletplastic.es/articulos.php?tipo=palots&subtipo=0

- Ministerio de Agricultura, Alimentación y Medio Ambiente (2012). Maquinaria Agricola
  http://www.magrama.gob.es/app/mecanizacion/fichamaquinaria.aspx?n1=8&n2=2&n3=0&n4=0

- Palet Plastic (2011). Catálogo de palots.
  http://www.paletplastic.es/articulos.php?tipo=palots&subtipo=0

7.2. Visión artificial

- JasVisio (2010). Visión artificial aplicada a la industria.
  http://www.jasvisio.com/aplicaciones-vision-artificial-industria.html

- Faico, Contro de innovación y tecnología (2011). Área de tecnologías de visión artificial.
  http://www.faico.org/seccion/visionartificial/

- Universidad de Córdoba; Prof. Dr. Fernández N. L. (2011). Introducción a la visión artificial. Visión Artificial Avanzada
  Disponible en:
  http://www.uco.es/users/ma1fegan/2011-2012/vision/Temas/Vision-artificial.pdf

7.3. Sistemas de iluminación

- Así Funciona, SL (2004). Que es un LED
  http://www.asifunciona.com/fisica/ke_led/ke_led_5.htm

- Ofiled (2013). Ventajas de la iluminación LED
  http://www.ofiled.com/led/ventajas-de-la-iluminacion-led

- Led Planet, S.L. (2007). Las 5 Ventajas LED
  http://www.ofiled.com/led/ventajas-de-la-iluminacion-led

- TodaCultura (2009). Luz Infrarroja
  http://www.todacultura.com/glosarioacuarela/luzinfrarroja.htm

- Spitzer (2009). ¿Qué es el infrarrojo?
  http://legacy.spitzer.caltech.edu/espanol//edu/ir/infrared.html

- Bachelor of Science in Public Health, BSPH; Prof. Colon de Jorge C. M. (2009). Aplicaciones de la radiación infrarroja.
  Disponible en:
  http://www.slideshare.net/naturopatia/aplicaciones-de-la-luz-infra-roja

7.4. Hardware

- Teledyne Dalsa (2013). Catálogo cámaras Genie
  http://www.teledynedalsa.com/imaging/products/cameras/area-scan/genie/

- Teledyne Dalsa (2013). Catálogo PC Geva
  http://www.teledynedalsa.com/imaging/products/va/

- Machine Building (2010). GEVA compact vision PC supports multiple cameras
  http://www.machinebuilding.net/p/p3539.htm

- Electan (2006). Solenoide / Electroimán
http://www.electan.com/datasheets/cebek/C-6092.pdf

- National Instruments Corporation (2008). Cómo hacer medidas con células de carga o transductores de presión
http://www.ni.com/white-paper/7138/es

- HBM España (2010). Células de carga de HBM: la solución correcta para cualquier aplicación
http://www.hbm.com/es/menu/productos/celulas-de-carga/

- B&R (2012). 4PP420.1043-75, Datasheet
http://www.br-automation.com/en-us/products/visualization-and-operation/power-panel-300400/power-panel-400-embedded/4pp4201043-75/

- B&R (2012). X20PS9400, Datasheet
http://www.br-automation.com/downloads_br_productcatalogue/BRP4440000000000000103125/X20PS9400-GER.pdf

- B&R (2012). X20DI4371, Datasheet
http://www.br-automation.com/downloads_br_productcatalogue/BRP4440000000000000100211/X20DI4371-GER.pdf

- B&R (2012). X20AI1744, Datasheet
http://www.br-automation.com/downloads_br_productcatalogue/BRP4440000000000000123269/X20AI1744-GER.pdf

- B&R (2012). X20AO4622, Datasheet
http://www.br-automation.com/downloads_br_productcatalogue/BRP4440000000000000104357/X20AO4622-ENG.pdf

- B&R (2012). X20DO8332, Datasheet
  http://www.br-automation.com/downloads_br_productcatalogue/BRP4440000000000000168312/X20DO8332-ENG.pdf

- B&R (2012). X20PS2100, Datasheet
  http://www.br-automation.com/downloads_br_productcatalogue/BRP4440000000000000104445/X20PS2100-ENG.pdf

7.5. Software

- Teledyne Dalsa (2013). Sherlock Vision System Software
  http://www.teledynedalsa.com/imaging/products/software/sherlock/

- Stemmer Imaging, Ltd. (2013). DALSA Sherlock - Machine vision software for industry
  http://www.stemmer-imaging.co.uk/en/products/line/dalsa.sherlock

- B&R (2012). Automation Studio
  http://www.br-automation.com/en/products/software/

7.6. Sistemas de comunicación

- Galeon (2011). Protocolo TCP/IP
  http://protocolotcpip.galeon.com/

- Microsoft (2013). Componentes de red y protocolos TCP/IP de nueva generación
  http://technet.microsoft.com/es-es/library/cc754287(v=ws.10).aspx

8. ANEXOS

8.1. ANEXO 1: Esquemas eléctricos

8.2. ANEXO 2: Planos mecánicos

8.2.1. Soportes cámaras

8.2.2. Soportes barras de LED

8.2.3. Estructura campana cámaras

8.2.4. Conjunto campana cámaras

8.2.5. Conjunto caja pantalla PLC A

8.2.6. Conjunto caja pantalla PLC B

Universitat de Vic
Escola Politècnica Superior

ANEXOS

ANEXO 1: Esquemas eléctricos

ANEXO 2: Planos mecánicos

Soportes cámaras

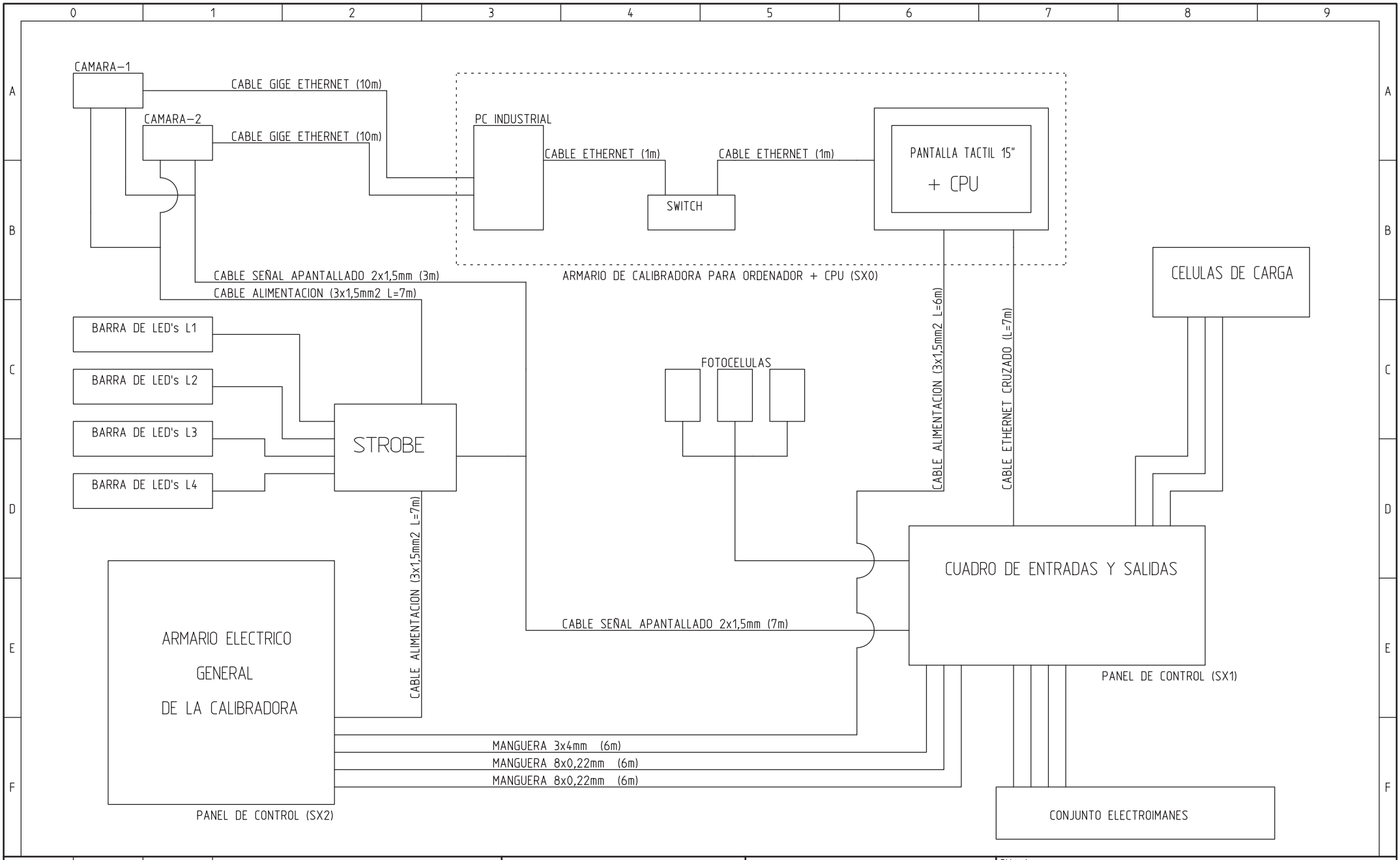
Soportes barras de LED

Estructura campana cámaras

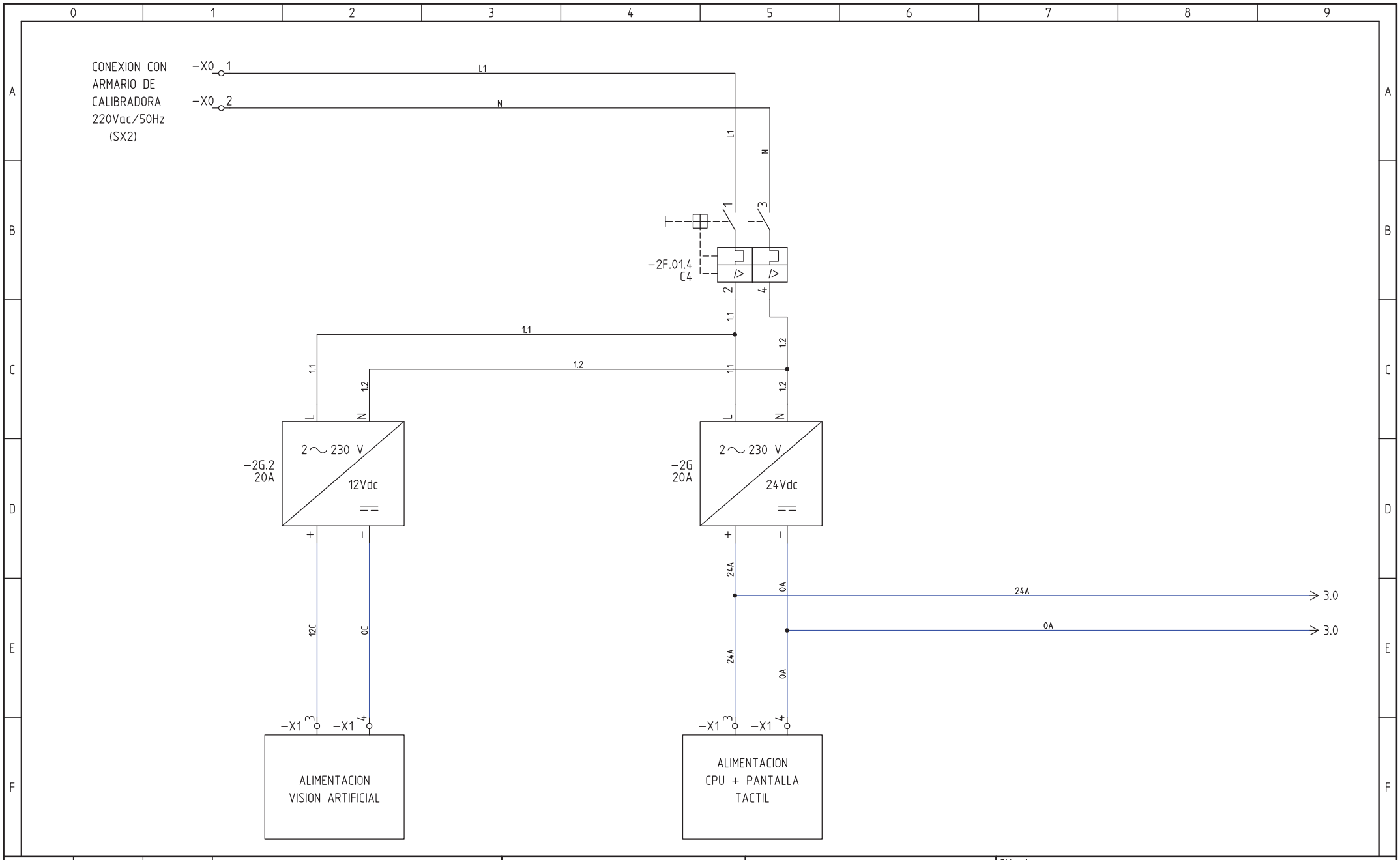
Conjunto campana cámaras

Conjunto caja pantalla PLC A

Conjunto caja pantalla PLC B



	Fecha	Nombre	Proyecto:		Título Hoja:	Cliente:	
Dibujado	10/03/2010	ECL	S20–0100–0002e	ERICO CRUZ LEMUS	ESQUEMA DISTRUBUCION		TFC UVIC
Revisado	10/03/2010		ELECTRONICA CALIBRADORA		CONEXIONADO	Documento Num:	Esquema Tipo: Esquemas de circuitos / Pag. 1. / Total Pag. 10
Modificado	10/03/2010		SXC3V26				



	Fecha	Nombre
Dibujado	10/02/2010	ECL
Revisado	10/02/2010	
Modificado	04/06/2009	

Proyecto:
S20-0100-0002e
ELECTRONICA CALIBRADORA
SXC3V26

ERICO CRUZ LEMUS

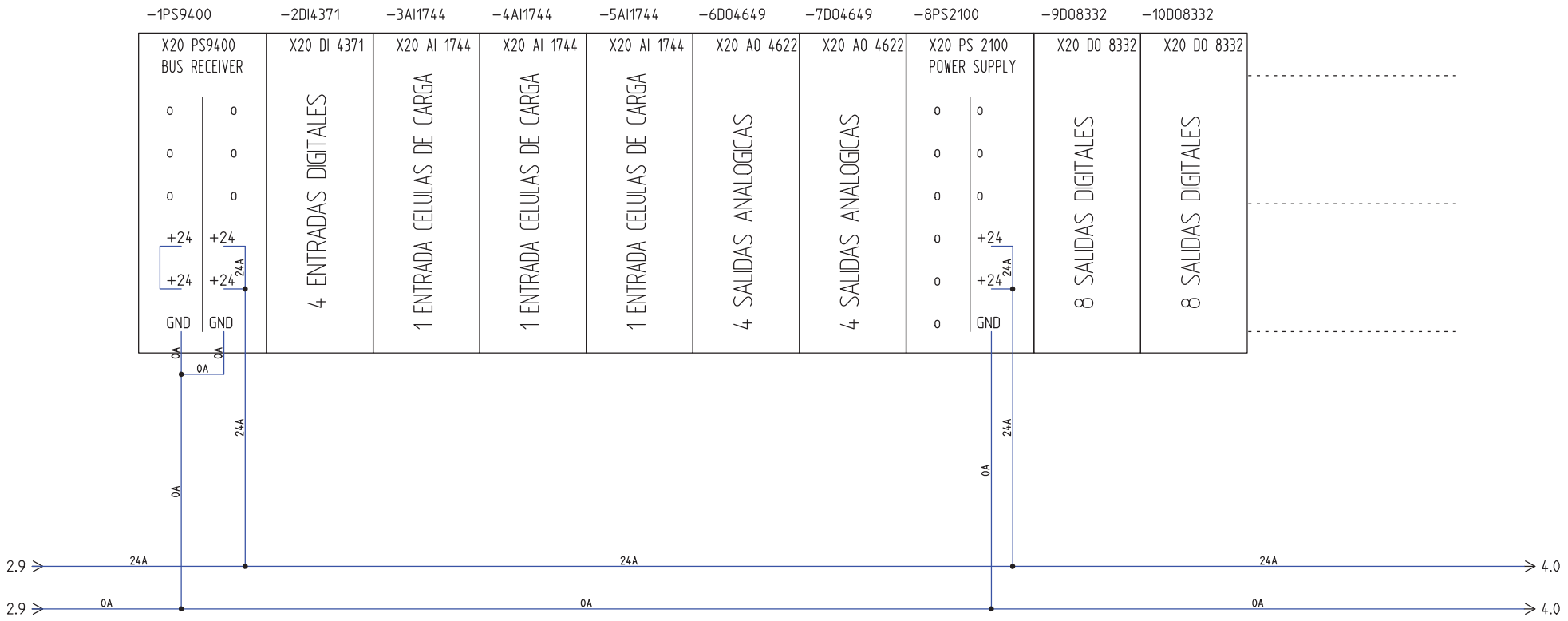
Título Hoja:
ALIMENTACION
DENTRO DE ARMARIO ELECTRICO (SX2)
PRINCIPAL DE CALIBRADORA

Cliente:
TFC UVIC

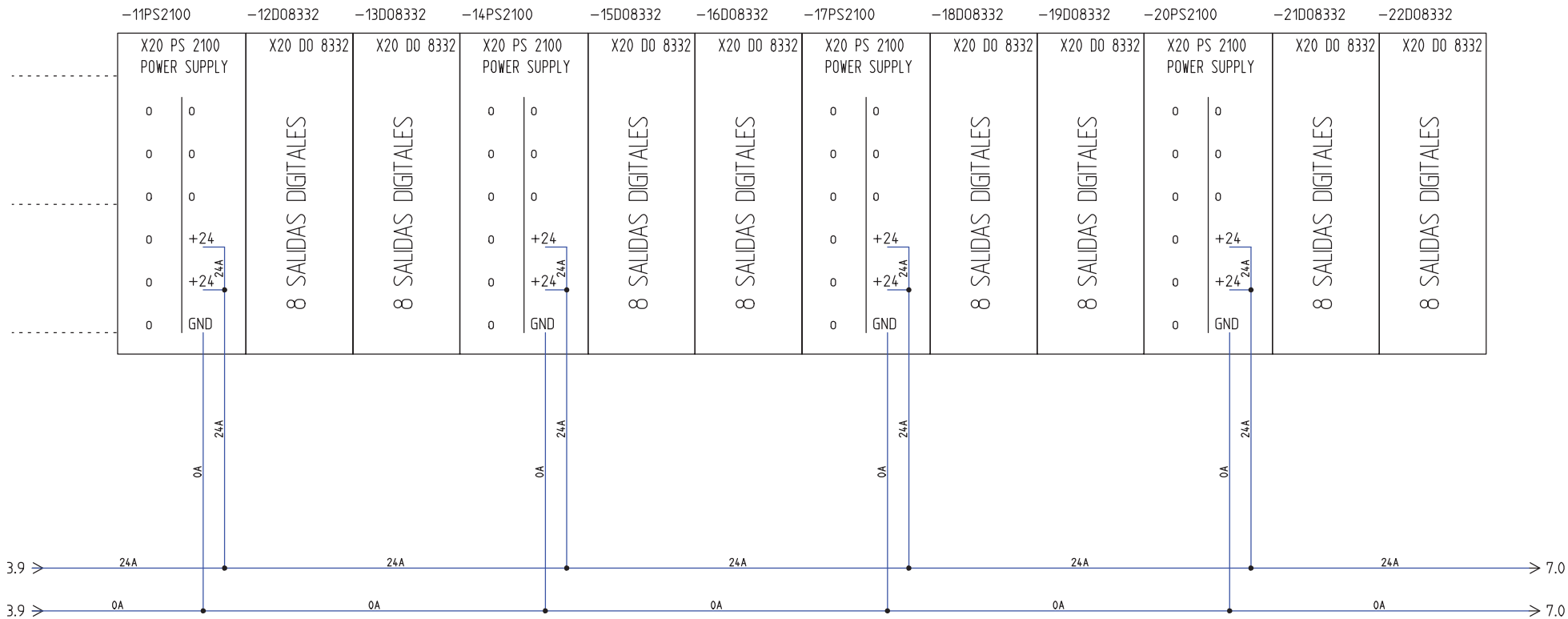
Documento Num:

Esquema Tipo:
Esquemas de circuitos

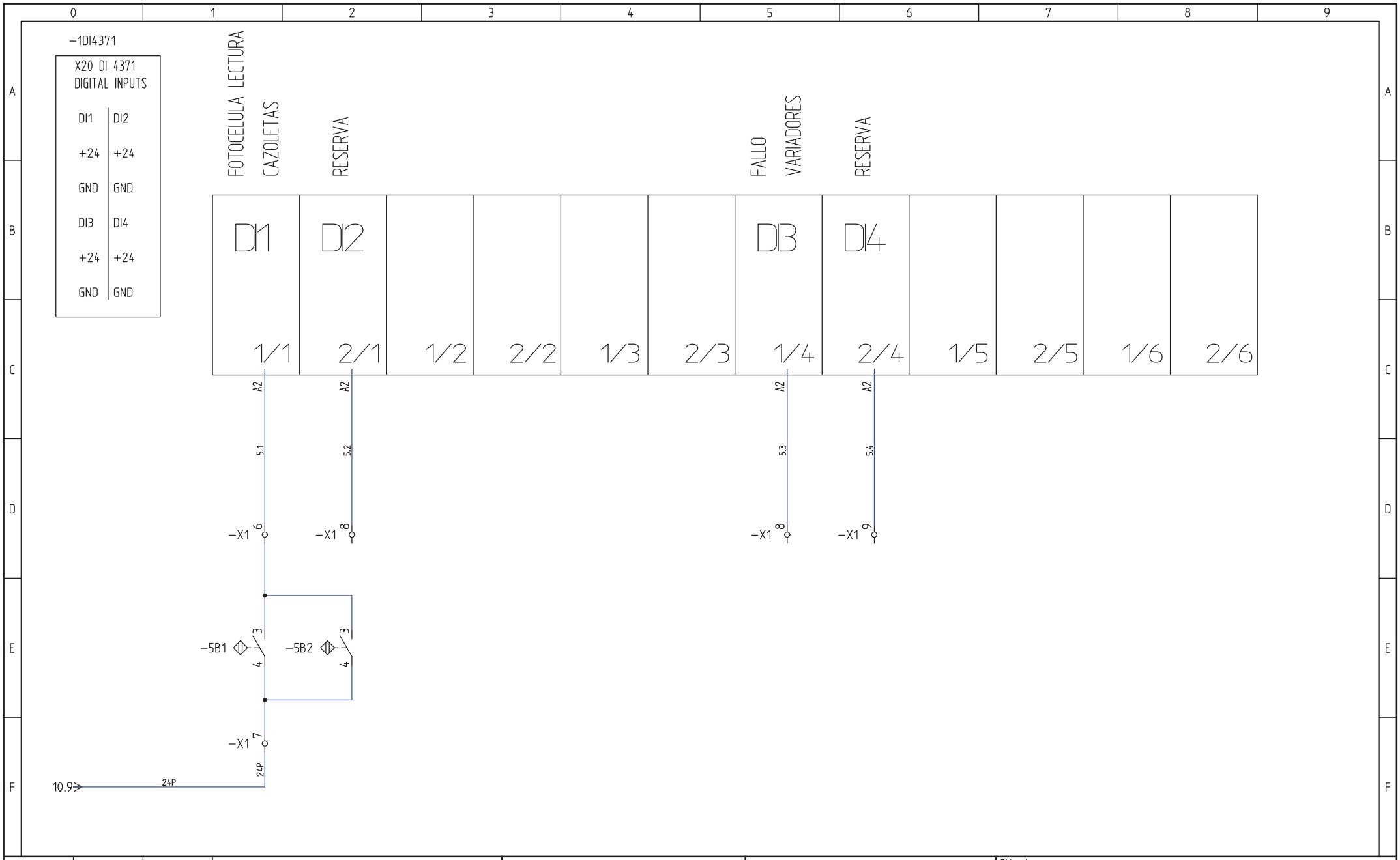
Pag. 2.
Total Pag. 10



	Fecha	Nombre	Proyecto:		Título Hoja:	Cliente:	
Dibujado	10/03/2010	ECL	S20−0100−0002e	ERICO CRUZ LEMUS	ESQUEMA DISTRIBUCION PLC 1	TFC UVIC	
Revisado	10/03/2010		ELECTRONICA CALIBRADORA		UNIDADES ENTRADAS/SALIDAS PLC	Documento Num:	Esquema Tipo: Esquemas de circuitos / Pag. 3. / Total Pag. 10
Modificado	10/03/2010		SXC3V26		CUADRO PRINCIPAL (SX1)		



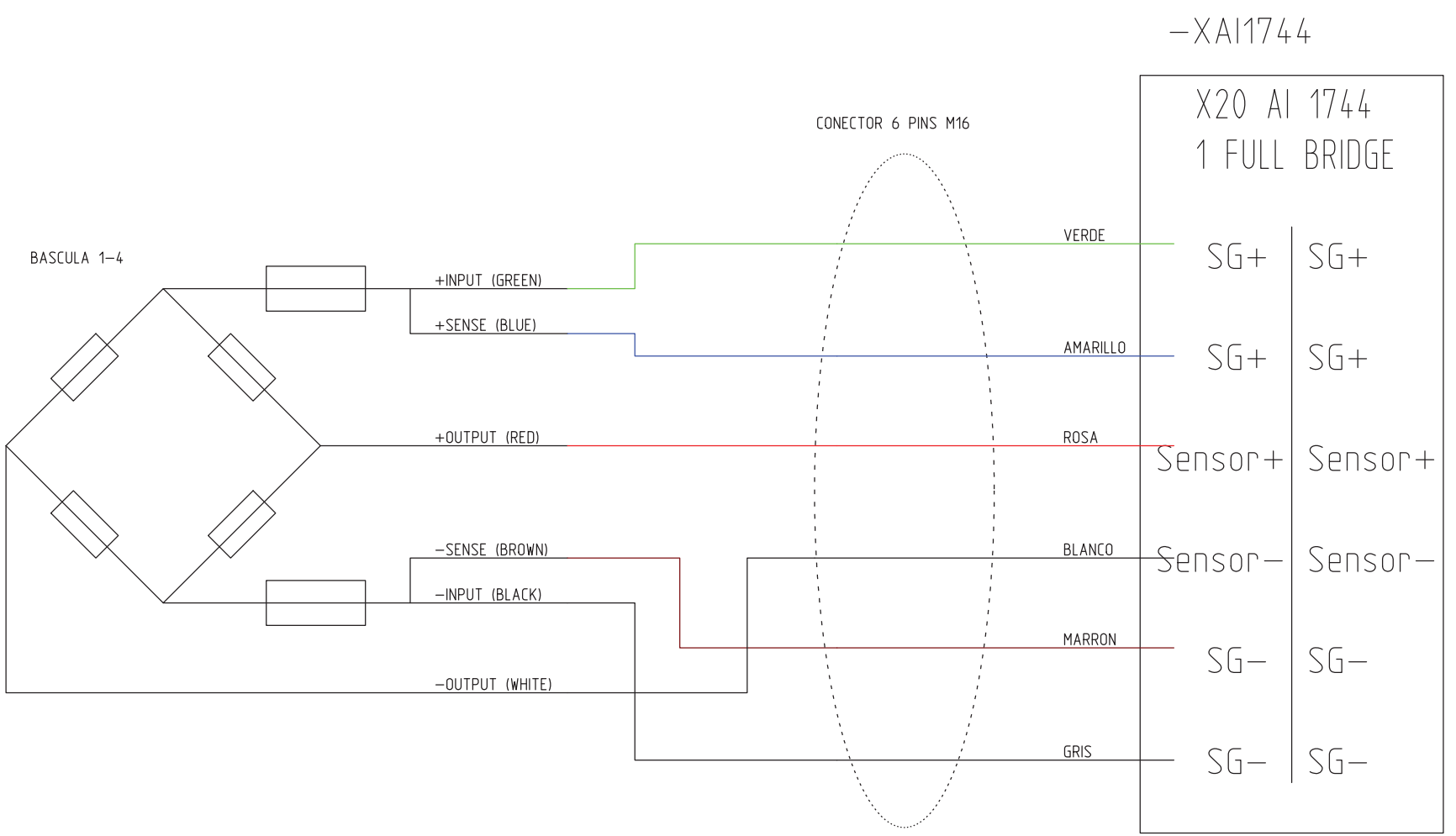
	Fecha	Nombre	Proyecto:		Título Hoja:	Cliente:	
Dibujado	10/03/2010	ECL	S20−0100−0002e	ERICO CRUZ LEMUS	ESQUEMA DISTRIBUCION PLC 2	TFC UVIC	
Revisado	10/03/2010		ELECTRONICA CALIBRADORA		UNIDADES ENTRADAS/SALIDAS PLC	Documento Num:	Esquema Tipo: Esquemas de circuitos / Pag. 4. / Total Pag. 10
Modificado	10/03/2010		SXC3V26		CUADRO PRINCIPAL (SX1)		



	Fecha	Nombre	Proyecto:		Título Hoja:	Cliente:	
Dibujado	10/03/2010	ECL	S20-0100-0002e	ERICO CRUZ LEMUS	ENTRADAS DIGITALES	TFC UVIC	
Revisado	10/03/2010		ELECTRONICA CALIBRADORA		CUADRO PRINCIPAL (SX1)	Documento Num:	Esquema Tipo: Esquemas de circuitos / Pag. 5. / Total Pag. 10
Modificado	10/03/2010		SXC3V26				

−XA1744

X20 AI 1744 1 FULL BRIDGE	
SG+	SG+
SG+	SG+
Sensor+	Sensor+
Sensor−	Sensor−
SG−	SG−
SG−	SG−



	Fecha	Nombre
Dibujado	10/03/2010	ECL
Revisado	10/03/2010	
Modificado	10/03/2010	

Proyecto:
S20−0100−0002e
ELECTRONICA CALIBRADORA
SXC3V26

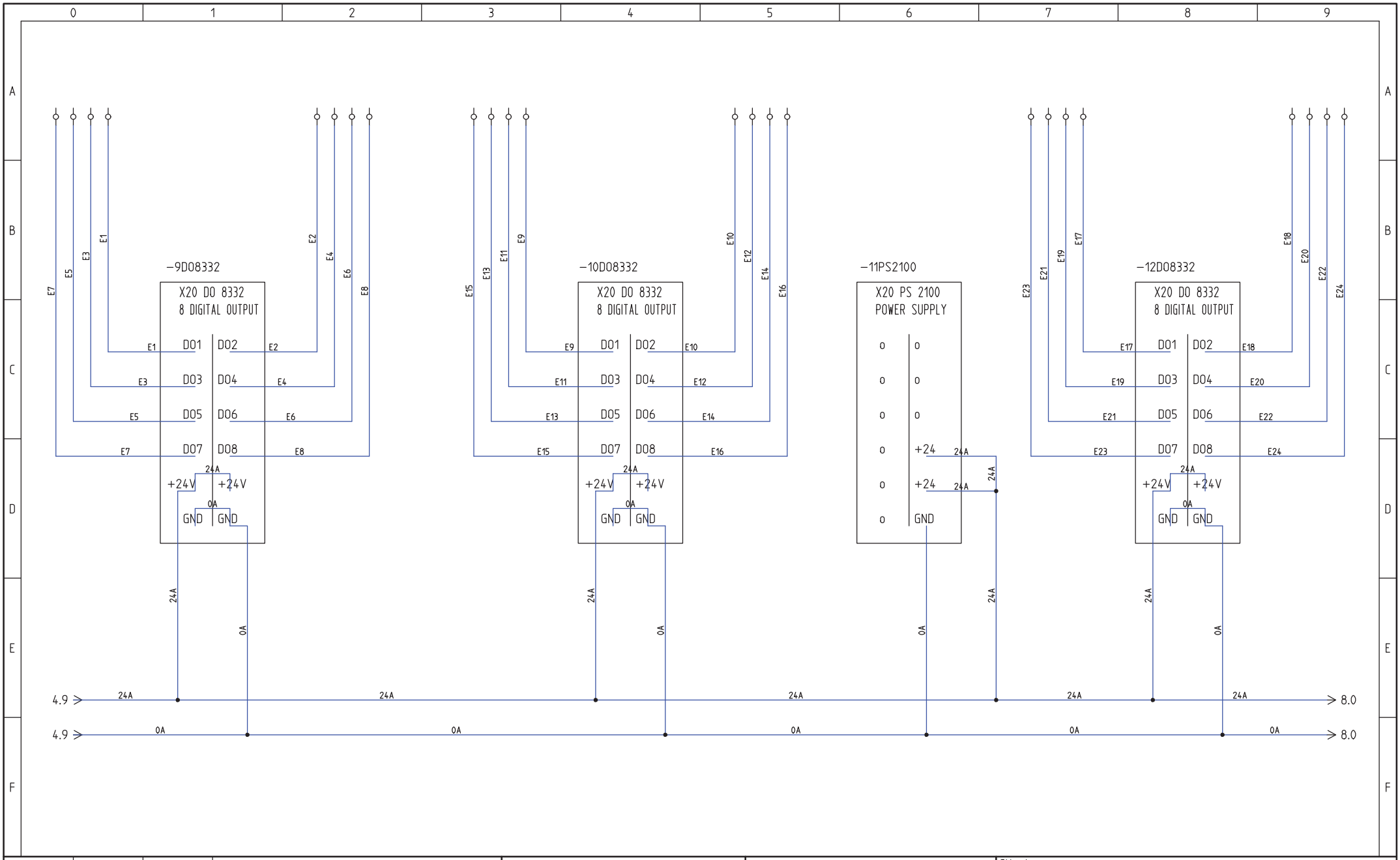
ERICO CRUZ LEMUS

Título Hoja:
ENTRADAS CELULAS DE CARGA
MODULO 1−4
CUADRO PRINCIPAL (SX1)

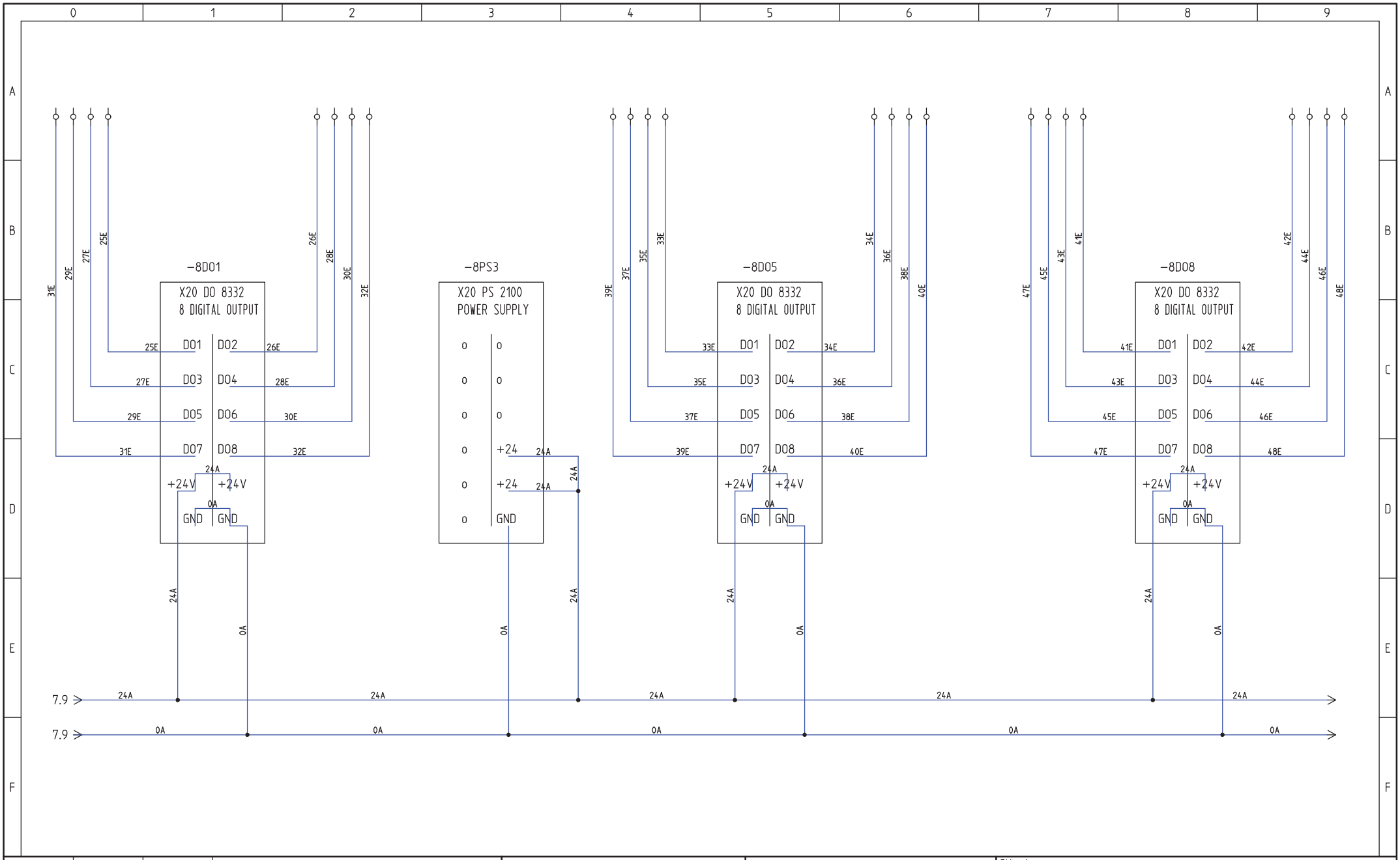
Cliente: TFC UVIC

Documento Num:

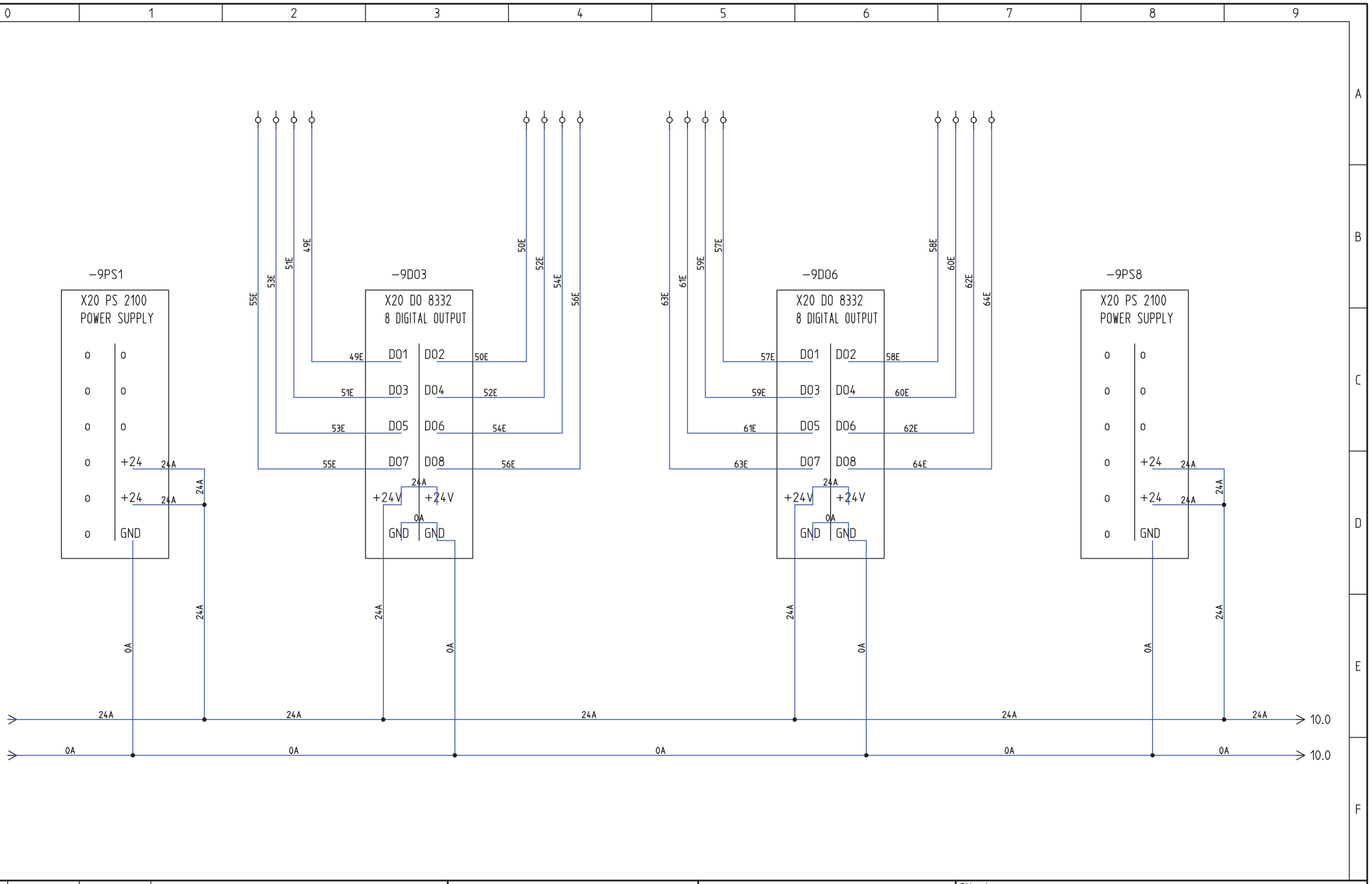
Esquema Tipo:
Esquemas de circuitos



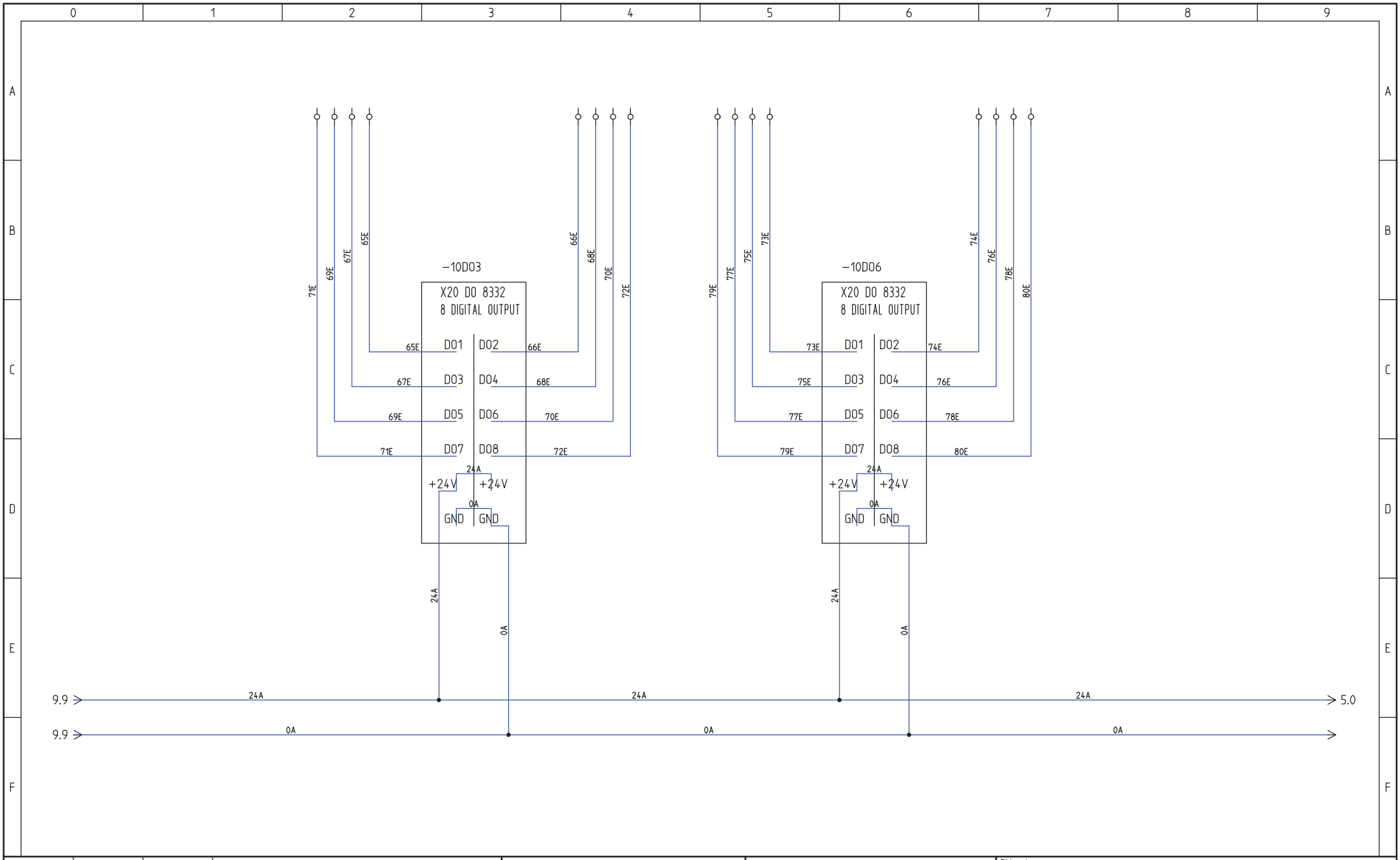
	Fecha	Nombre	Proyecto:		Título Hoja:	Cliente:	
Dibujado	10/03/2010	ECL	S20-0100-0002e	ERICO CRUZ LEMUS	SALIDAS RELE		TFC UVIC
Revisado	10/03/2010		ELECTRONICA CALIBRADORA		ELECTROIMANES 1-20	Documento Num:	Esquema Tipo: Esquemas de circuitos
Modificado	10/03/2010		SXC3V26		CUADRO PRINCIPAL (SX1)		Pag. 7. Total Pag. 10



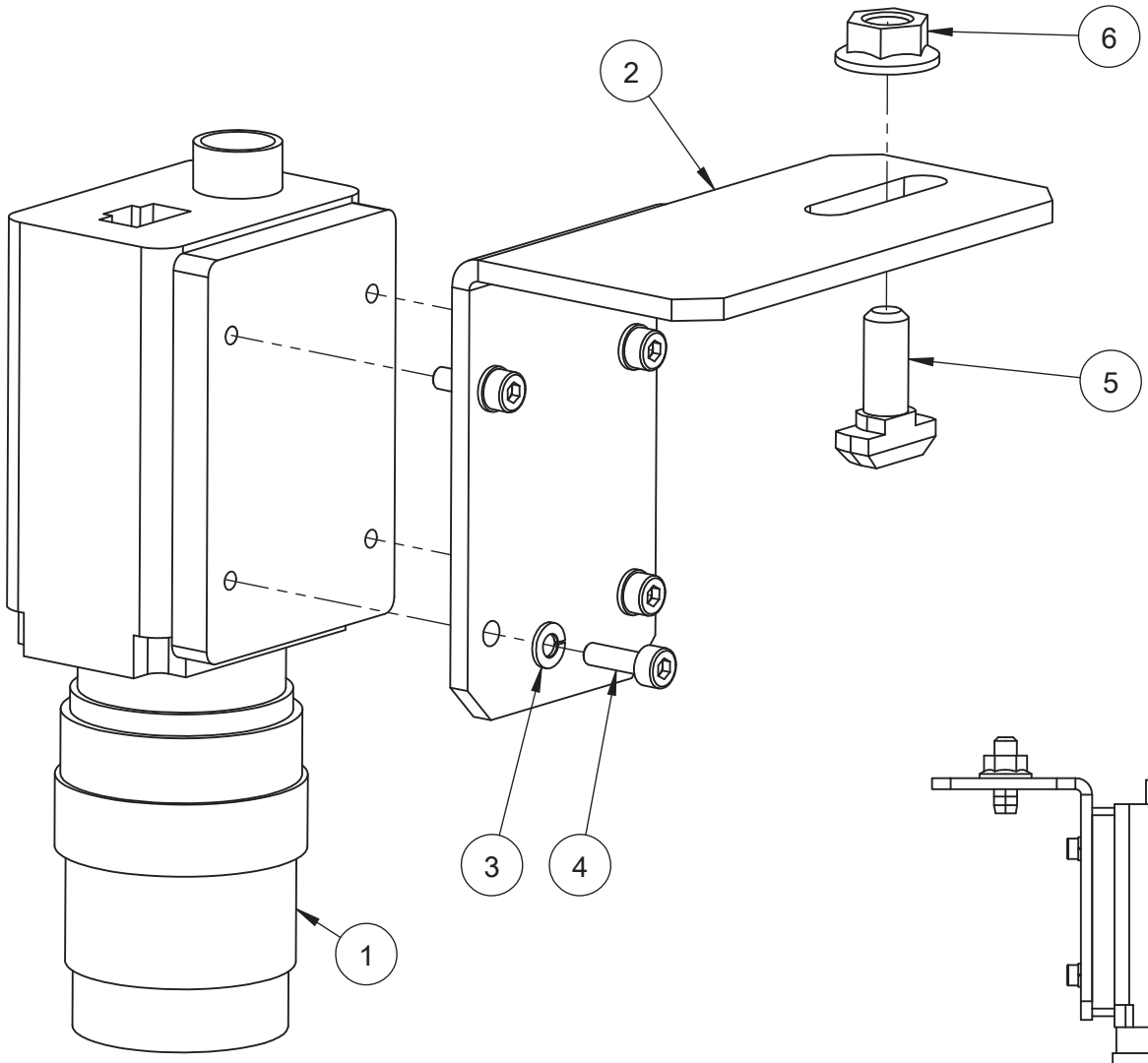
	Fecha	Nombre	Proyecto:	ERICO CRUZ LEMUS	Título Hoja:	Cliente:	TFC UVIC	
Dibujado	10/03/2010	ECL	S20-0100-0002e		SALIDAS RELE			
Revisado	10/03/2010		ELECTRONICA CALIBRADORA		ELECTROIMANES 20-40	Documento Num:	Esquema Tipo: Esquemas de circuitos	Pag. 8.
Modificado	10/03/2010		SXC3V26		CUADRO PRINCIPAL (SX1)			Total Pag. 10

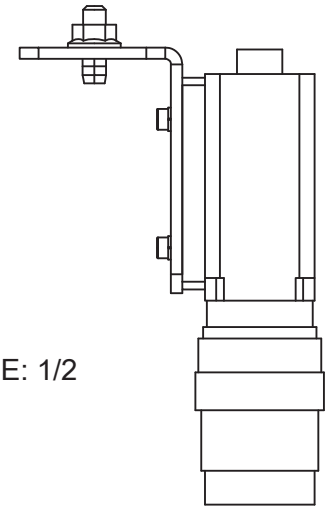


	Fecha	Nombre	Proyecto:		Título Hoja:	Cliente:	
Dibujado	10/03/2010	ECL	S20-0100-0002e	ERICO CRUZ LEMUS	SALIDAS RELE	TFC UVIC	
Revisado	10/03/2010		ELECTRONICA CALIBRADORA		ELECTROIMANES 40-60	Documento Num:	Esquema Tipo: Esquemas de circuitos / Pag. 9. / Total Pag. 10
Modificado	10/03/2010		SXC3V26		CUADRO PRINCIPAL (SX1)		



	Fecha	Nombre	Proyecto:	ERICO CRUZ LEMUS	Título Hoja:	Cliente: TFC UVIC	
Dibujado	14/02/2011	ECL	S20−0100−0002e		SALIDAS RELE		
Revisado	14/02/2011		ELECTRONICA CALIBRADORA		ELECTROIMANES 40−60	Documento Num:	Esquema Tipo: Esquemas de circuitos — Pag. 10. Total Pag. 10
Modificado	14/02/2011		SXC3V26		CUADRO PRINCIPAL (SX1)		





E: 1/2

Número	**Descripción**	**Código**	Cantidad
1	Cámara Genie M1024-1/3" Monocromo	E050000011	1
2	Soporte camara	s000174	1
3	Arandela de presión DIN 127B M3	T080000004	4
4	Tornillo DIN 912 M3x10	T040000026	4
5	Cabeza de martillo Bosch M6x25 R8	M100000021	1
6	Tuerca cab.martillo Bosch M6	M100000022	1

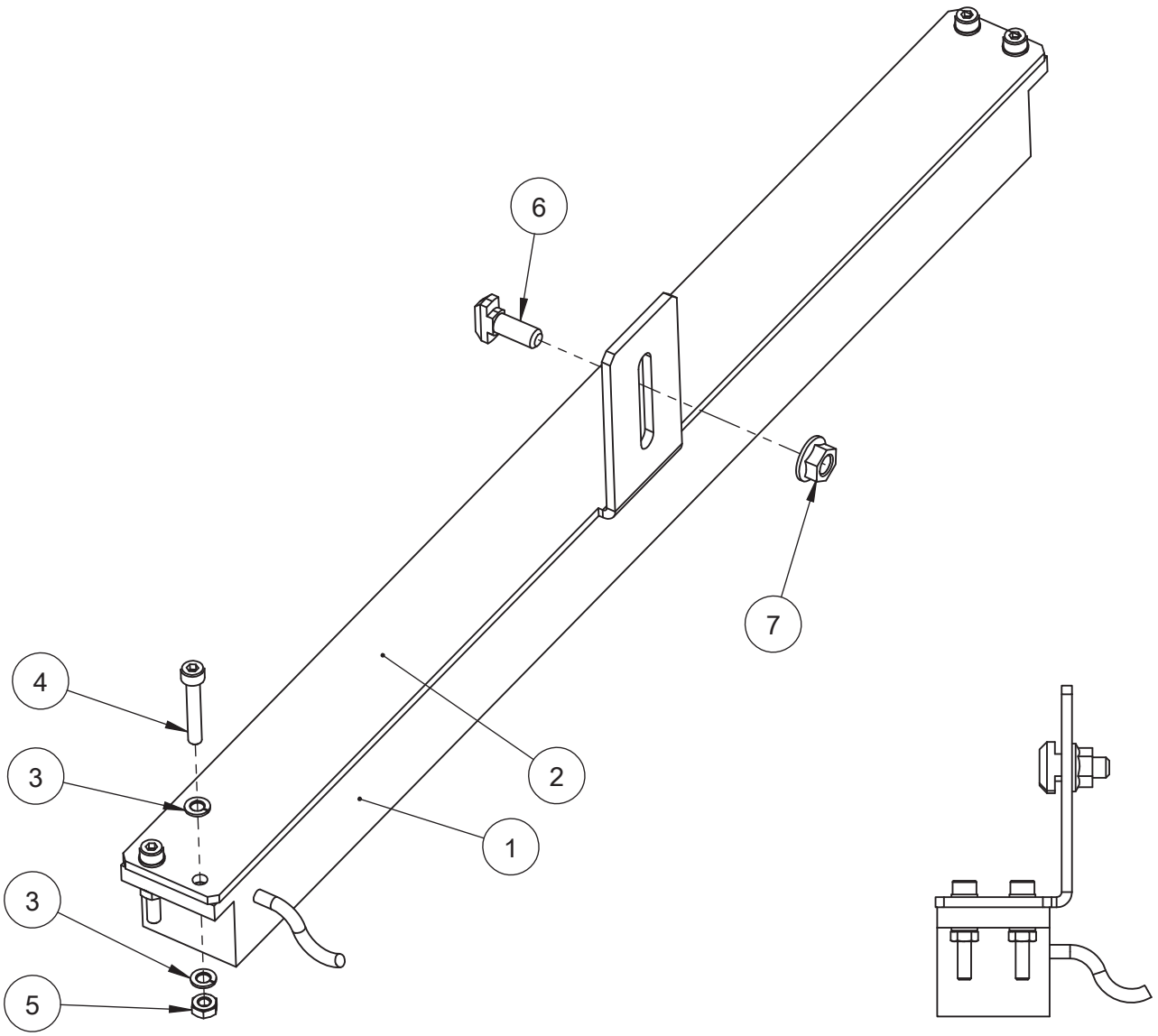
RUGOSIDAD Ra ISO/R 468	3,2-12,5	0,8-3,2	0,2-0,8	0,016-0,2	Diferencias admisibles para medidas sin indicacion de tolerancia	hasta 6	mas de 6 hasta 30	mas de 30 hasta 120	mas de 120 hasta 315	mas de 315 hasta 1000	mas de 1000 hasta 2000	mas de 2000 hasta 4000	mas de 4000 hasta 8000
SIGNOS A EXTINGUIR	▽	▽▽	▽▽▽	▽▽▽▽	DIN 7168 Precision media	±0,1	±0,2	±0,3	±0,5	±0,8	±1,2	±2	±3

PLANO DE REFERENCIA	N° SUSTITUIDO	NUEVO N°
-	-	-
FECHA-NOMBRE	-	-

Las medidas ☐ seran expres. verificadas.
Las medidas ( ) no tienen importancia.
Sin indicacion las esquinas seran a 0,5mm R.
Sin indicacion las aristas seran 0,5mm a 45°.

Sin indicacion expresa roscas de calidad 6 (media)
Tuercas 6H y Tornillos 6g.

ERICO CRUZ LEMUS	DESCRIPCIÓN	ANEXO 2	
	CONJUNTO	8.2.1. Soportes Cámaras	
	MAQUINA	TFC UVIC	
DIBUJADO 21/09/2010 E. Cruz	Hoja 1 de 1	NUMERO DE ARTICULO	ESCALA 1:1
VERIFICADO			Nº PLANO e000175
			ORDEN MOD.



Número	**Descripción**	**Código**	Cantidad
1	Barra LEDs infra. IR400x25mm	E050000012	1
2	Soporte leds	s000175	1
3	Arandela de presión din 127 B M4	T080000003	8
4	Tornillo DIN 912 M4x25	T040000021	4
5	Tuerca DIN 934 M4	T030000008	4
6	Cabeza de martillo Bosch M6x25 R8	M100000021	1
7	Tuerca cab.martillo Bosch M6	M100000022	1

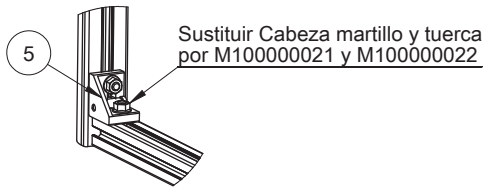
RUGOSIDAD Ra ISO/R 468	3,2-12,5	0,8-3,2	0,2-0,8	0,016-0,2	Diferencias admisibles para medidas sin indicacion de tolerancia	hasta 6	mas de 6 hasta 30	mas de 30 hasta 120	mas de 120 hasta 315	mas de 315 hasta 1000	mas de 1000 hasta 2000	mas de 2000 hasta 4000	mas de 4000 hasta 8000
SIGNOS A EXTINGUIR	▽	▽▽	▽▽▽	▽▽▽▽	DIN 7168 Precision media	±0,1	±0,2	±0,3	±0,5	±0,8	±1,2	±2	±3

PLANO DE REFERENCIA	N° SUSTITUIDO	NUEVO N°
-	-	-
FECHA-NOMBRE	-	-

Las medidas ☐ seran expres. verificadas.
Las medidas ( ) no tienen importancia.
Sin indicacion las esquinas seran a 0,5mm R.
Sin indicacion las aristas seran 0,5mm a 45°.

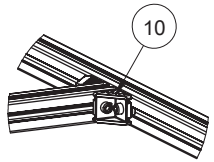
Sin indicacion expresa roscas de calidad 6 (media)
Tuercas 6H y Tornillos 6g.

ERICO CRUZ LEMUS			DESCRIPCIÓN	ANEXO 2		
			CONJUNTO	8.2.2. Soportes Barras de LED		
	FECHA	NOMBRE	MAQUINA	TFC UVIC		
DIBUJADO	21/09/2010	E.Cruz	Hoja 1 de 1	NUMERO DE ARTICULO	ESCALA 1:2	Nº PLANO e000176	ORDEN MOD.
VERIFICADO						

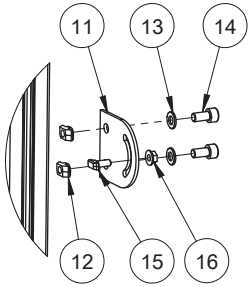


Union perfiles 45x45 con 30x30

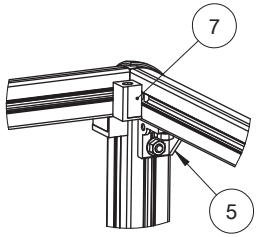
DETALLE E
ESCALA 1 : 5



DETALLE D
ESCALA 1 : 5

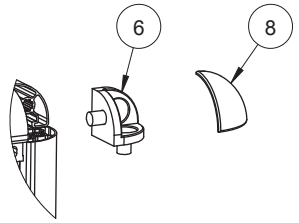


DETALLE C
ESCALA 1 : 5

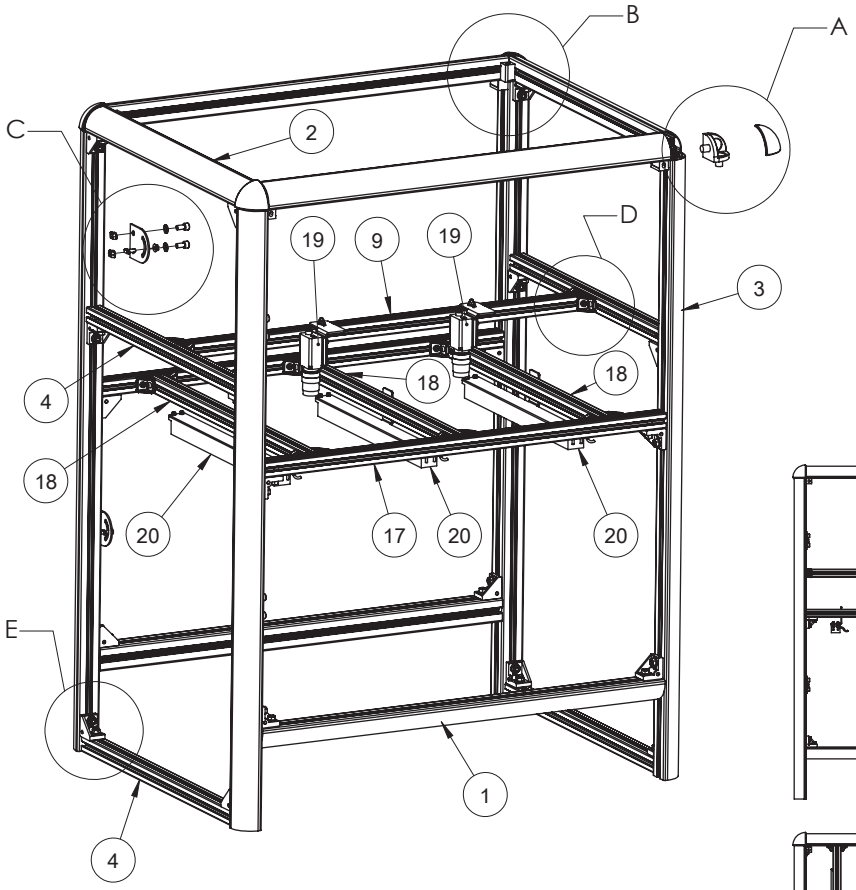


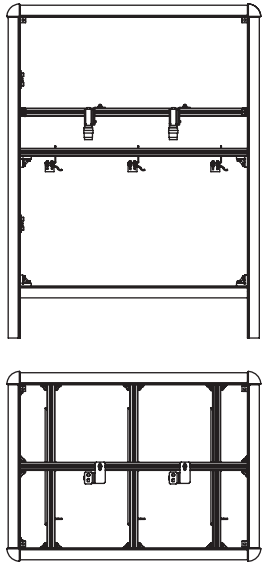
Pieza nº7 fijación reynobond

DETALLE B
ESCALA 1 : 5



DETALLE A
ESCALA 1 : 5





E= 1:5

Número	Descripción	Código	Cantidad
1	Perfil Bosch 45X45L R L=850mm	M100000010	4
2	Perfil Bosch 45x45L R L=610mm	M100000011	2
3	Perfil Bosch 45x45L R L=1200mm	M100000012	4
4	Perfil Bosch 30x30 L=610mm	M100000013	4
5	Escuadra.Bosch Variofix-block S40/45 M8x25 R10	M100000018	20
6	Rinconera esferica Bosch 45x45	M100000016	4
7	Bosch Variofix-block S40/45	M100000019	8
8	Tapa rinconera esferica Bosch 45x45	M100000017	4
9	Perfil Bosch 30x30 L=865mm	M100000014	1
10	Escuadra Bosch 30x30 M6x14 R8	M100000020	16
11	Platina union perfiles armario	s000080	4
12	Tuerca de martillo M8 R10	M100000001	8
13	Arandela DIN 125-B M8	T010000004	8
14	Tornillo DIN 912 M8x16	T040000008	8
15	Cabeza de martillo Bosch M6x25 R8	M100000021	4
16	Tuerca cab.martillo Bosch M6	M100000022	4
17	Perfil Bosch 30x30 R8 L=850mm	M100000025	2
18	Perfil Bosch 30x30 R8 L=625mm	M100000026	3
19	Conjunto Cam. Infaimon + soporte	s000086	2
20	Conjunto iluminación Infaimon + soporte	s000087	3

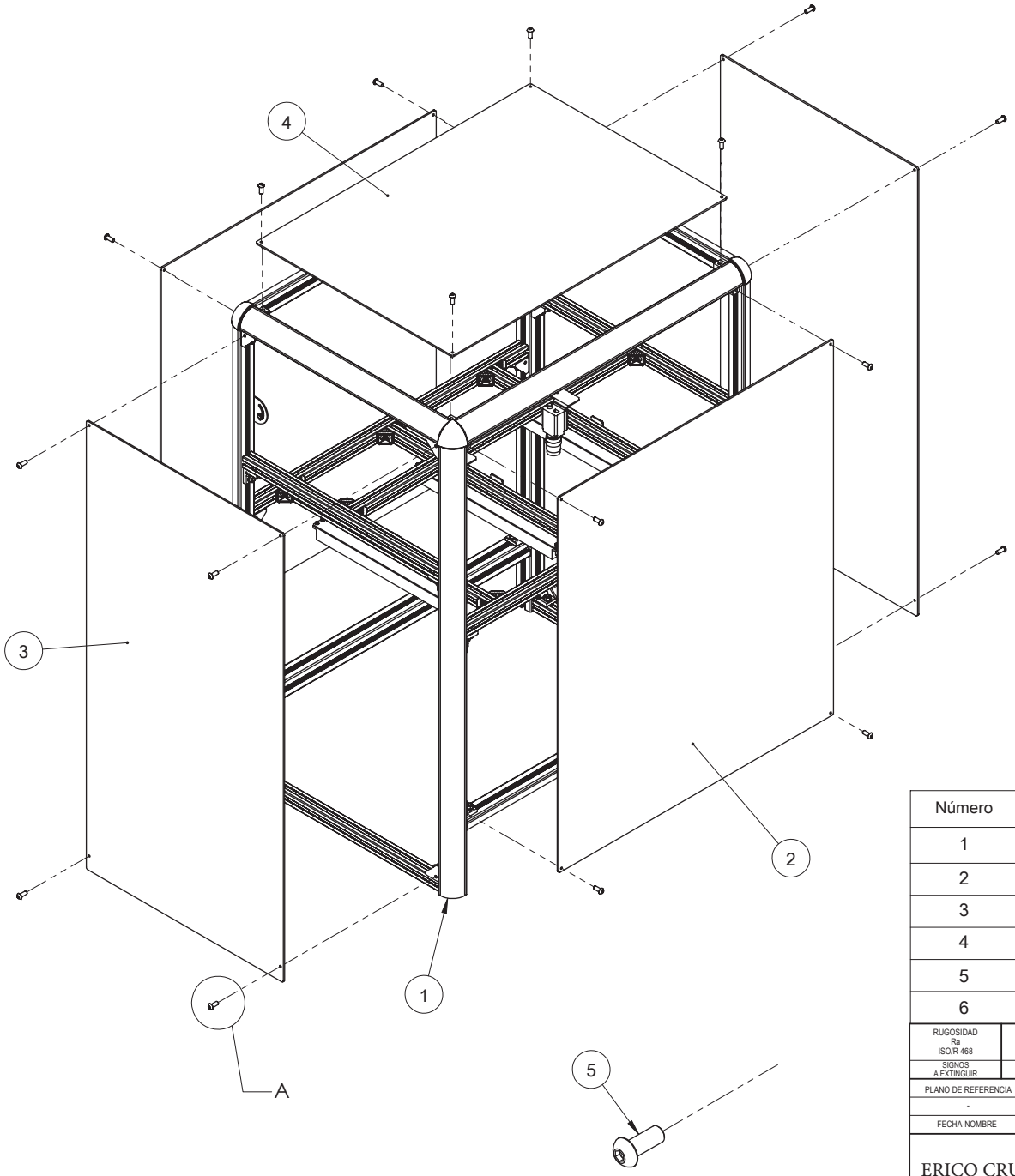
RUGOSIDAD Ra ISO/R 468	3,2-12,5	0,8-3,2	0,2-0,8	0,016-0,2	Diferencias admisibles para medidas sin indicacion de tolerancia	hasta 6	mas de 6 hasta 30	mas de 30 hasta 120	mas de 120 hasta 315	mas de 315 hasta 1000	mas de 1000 hasta 2000	mas de 2000 hasta 4000	mas de 4000 hasta 8000
SIGNOS A EXTINGUIR	▽	▽▽	▽▽▽	▽▽▽▽	DIN 7168 Precision media	±0,1	±0,2	±0,3	±0,5	±0,8	±1,2	±2	±3

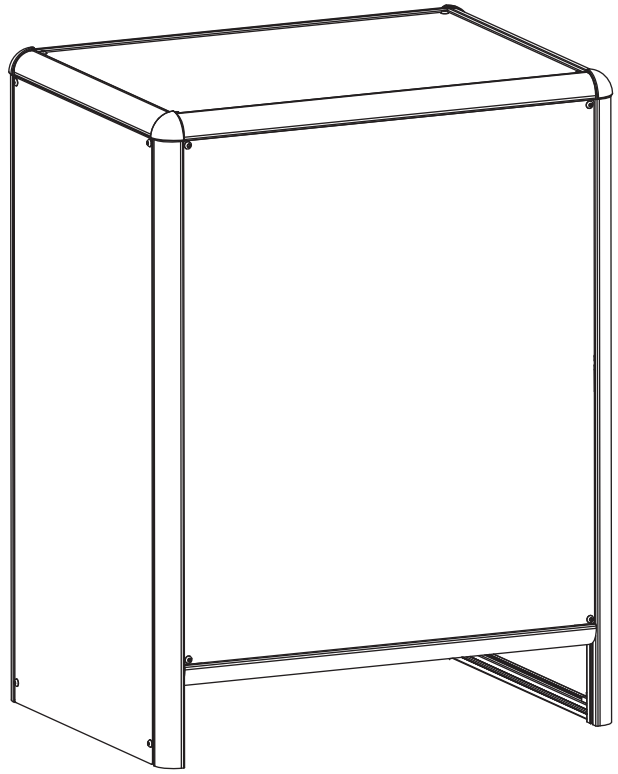
PLANO DE REFERENCIA	Nº SUSTITUIDO	NUEVO Nº
-	-	-
FECHA-NOMBRE	-	-

Las medidas [ ] seran expres. verificadas.
Las medidas ( ) no tienen importancia.
Sin indicacion las esquinas seran a 0,5mm R.
Sin indicacion las aristas seran 0,5mm a 45°.

Sin indicacion expresa roscas de calidad 6 (media)
Tuercas 6H y Tornillos 6g.

ERICO CRUZ LEMUS	DESCRIPCIÓN	ANEXO 2	
	CONJUNTO	8.2.3. Estructura Campana Cámaras	
	MAQUINA	TFC UVIC	
DIBUJADO 02/02/2010 E.Cruz	Hoja 1 de 1	NUMERO DE ARTICULO	ESCALA 1:20
VERIFICADO			Nº PLANO e000177
			ORDEN MOD.





DETALLE A
ESCALA 1 : 2

Número	Descripción	Código	Cantidad
1	Conj. estructura campana camaras visión 4 lineas	s000088	1
2	Carenado 1 caja de camaras	s000083	2
3	Carenado 2 caja de camaras	s000084	2
4	Carenado 3 caja de camaras	s000085	1
5	Tornillo ISO 7380 M8x20	T060000002	19
6	Tornillo ISO 7380 M8x20	T060000002	1

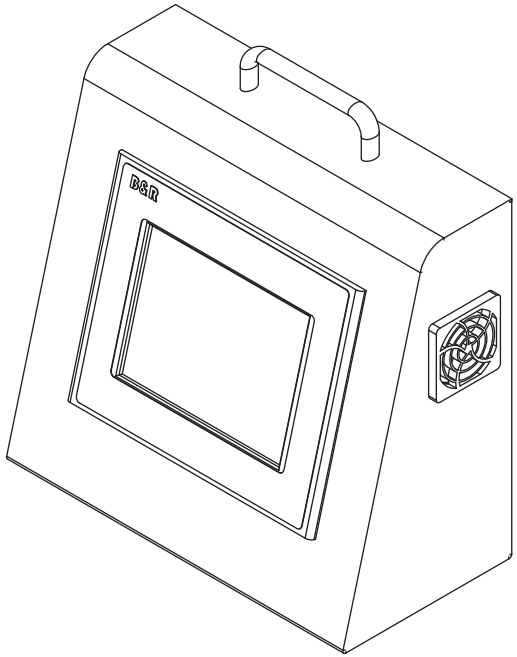
RUGOSIDAD Ra ISO/R 468	3,2-12,5	0,8-3,2	0,2-0,8	0,016-0,2	Diferencias admisibles para medidas sin indicacion de tolerancia	hasta 6	mas de 6 hasta 30	mas de 30 hasta 120	mas de 120 hasta 315	mas de 315 hasta 1000	mas de 1000 hasta 2000	mas de 2000 hasta 4000	mas de 4000 hasta 8000
SIGNOS A EXTINGUIR	▽	▽▽	▽▽▽	▽▽▽▽	DIN 7168 Precision media	±0,1	±0,2	±0,3	±0,5	±0,8	±1,2	±2	±3

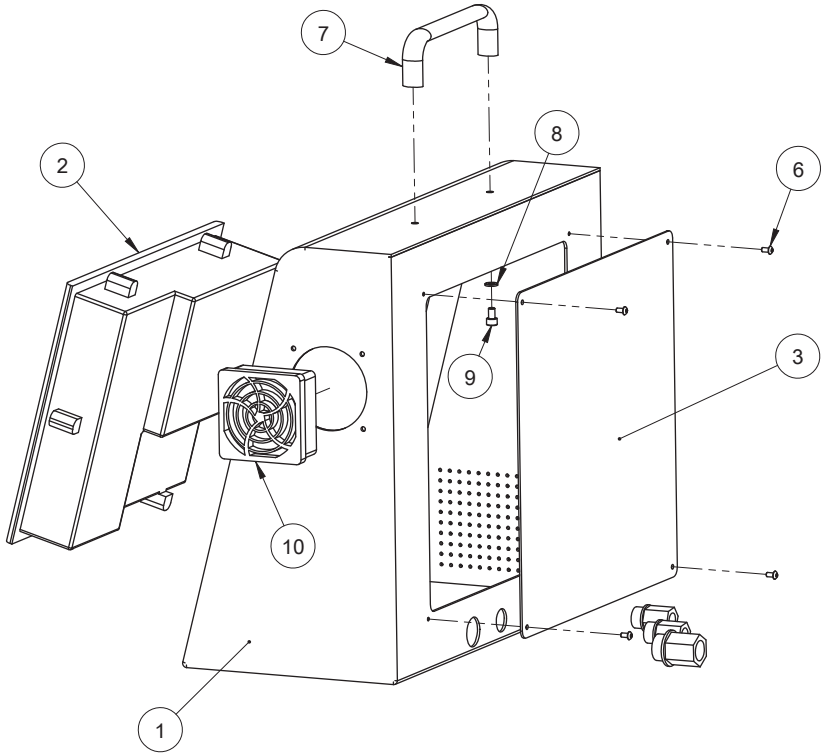
PLANO DE REFERENCIA	N° SUSTITUIDO	NUEVO N°
-	-	-
FECHA-NOMBRE	-	-

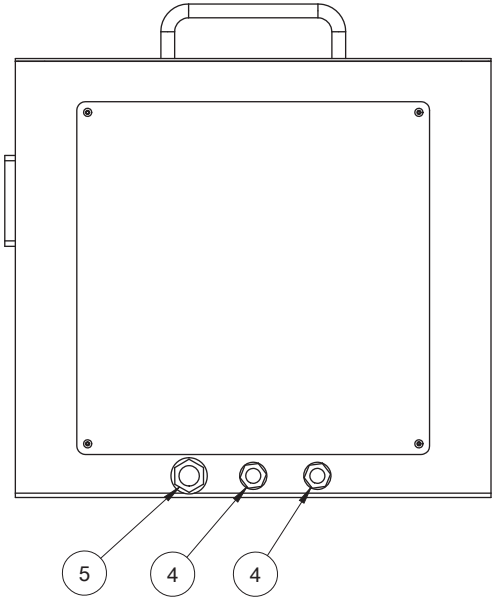
Las medidas ☐ seran expres. verificadas.
Las medidas ( ) no tienen importancia.
Sin indicacion las esquinas seran a 0,5mm R.
Sin indicacion las aristas seran 0,5mm a 45°.

Sin indicacion expresa roscas de calidad 6 (media)
Tuercas 6H y Tornillos 6g.

ERICO CRUZ LEMUS	DESCRIPCIÓN	ANEXO 2	
	CONJUNTO	8.2.4. Conjunto Campana Cámaras	
	MAQUINA	TFC UVIC	
DIBUJADO 02/02/2010 E.Cruz	Hoja 1 de 1	NUMERO DE ARTICULO	ESCALA 1:10
VERIFICADO			Nº PLANO e000178
			ORDEN MOD.







Número	Descripción	Código	Cantidad
1	Caja pantalla B&R PP420 10.4"	s000091	1
2	Power Panel B&R PP420 10.4"	E040000021	1
3	Tapa caja pantalla B&R PP420 10.4"	s000092	1
4	Montage TEG	Racord electric M20x1.5	2
5	Montage TEG	Racord electric M25x1.5	1
6	Tornillo ISO 7380 M4x8	T060000007	4
7	Emka Asa 160B20 Aluminio anodizado	M090000013	1
8	Arandela DIN 125-B M6	T010000003	2
9	Tornillo DIN 912 M6x10	T040000017	2
10	Montage TEG	Ventilador	1

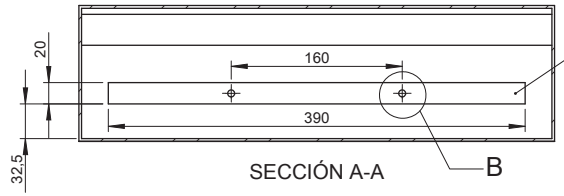
RUGOSIDAD Ra ISO/R 468	3,2-12,5	0,8-3,2	0,2-0,8	0,016-0,2	Diferencias admisibles para medidas sin indicacion de tolerancia	hasta 6	mas de 6 hasta 30	mas de 30 hasta 120	mas de 120 hasta 315	mas de 315 hasta 1000	mas de 1000 hasta 2000	mas de 2000 hasta 4000	mas de 4000 hasta 8000
SIGNOS A EXTINGUIR	▽	▽▽	▽▽▽	▽▽▽▽	DIN 7168 Precision media	±0,1	±0,2	±0,3	±0,5	±0,8	±1,2	±2	±3

PLANO DE REFERENCIA	N° SUSTITUIDO	NUEVO N°
-	-	-
FECHA-NOMBRE	-	-

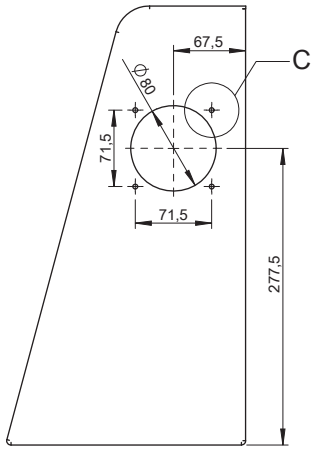
Las medidas [ ] seran expres. verificadas.
Las medidas ( ) no tienen importancia.
Sin indicacion las esquinas seran a 0,5mm R.
Sin indicacion las aristas seran 0,5mm a 45°.

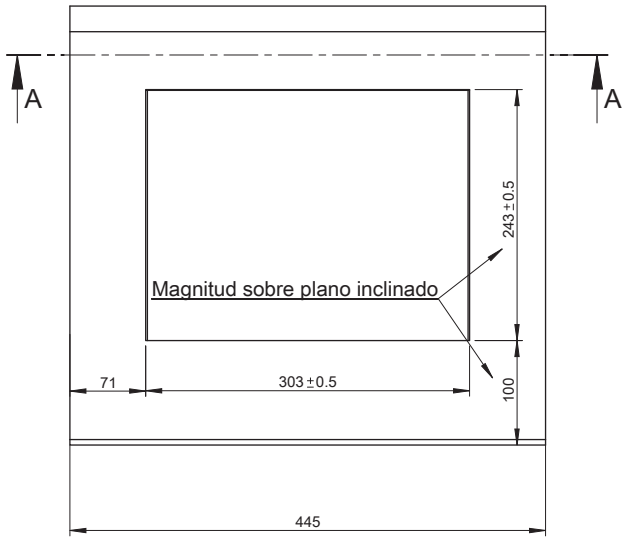
Sin indicacion expresa roscas de calidad 6 (media)
Tuercas 6H y Tornillos 6g.

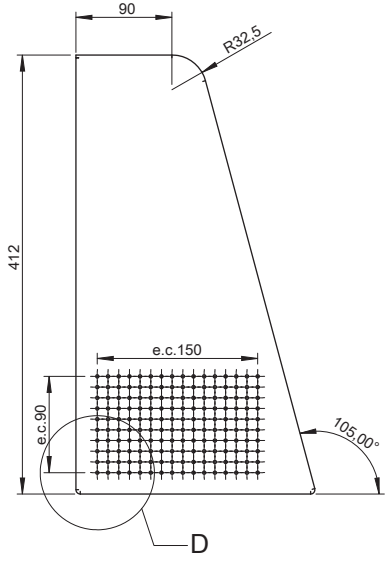
ERICO CRUZ LEMUS	DESCRIPCIÓN	ANEXO 2	
	CONJUNTO	8.2.5. Conjunto Caja pantalla PLC A	
	MAQUINA	TFC UVIC	
DIBUJADO 01/02/2010 E. Cruz	Hoja 1 de 1	NUMERO DE ARTICULO	ESCALA 1:10
VERIFICADO			Nº PLANO e000179
			ORDEN MOD.

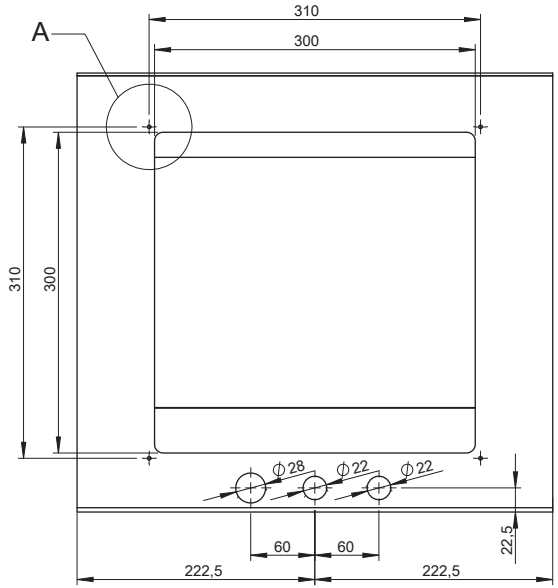


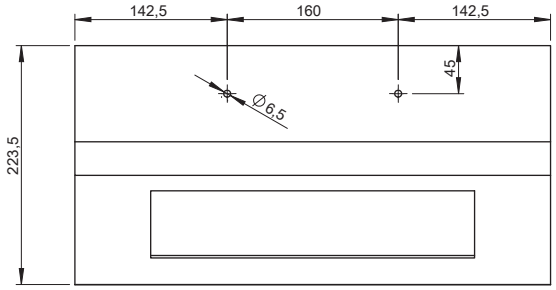
SECCIÓN A-A

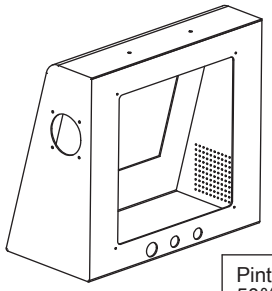


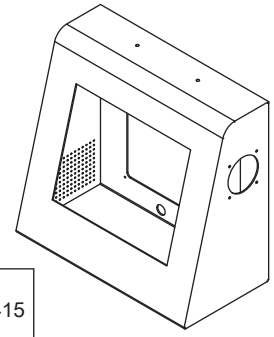




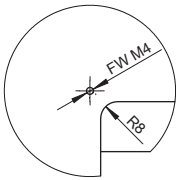






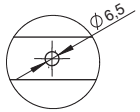


Pintura:
50% Gris blau pantone 5415
50% Blanc RAL 9010

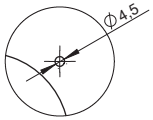


FW: Taladro flow drill

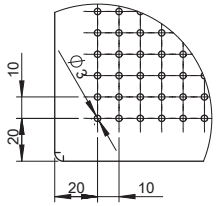
DETALLE A
ESCALA 2 : 5



DETALLE B
ESCALA 2 : 5



DETALLE C
ESCALA 2 : 5



Taladro diam.3mm (16 columnas x 10 filas)

DETALLE D
ESCALA 2 : 5

Material	Acabado	Fabricación	Peso	Cantidad
1023 Chapa de acero al carbono (SS)	Pintado	Plancha 2.5mm	9.5 kg	1

RUGOSIDAD Ra ISO/R 468	3,2-12,5	0,8-3,2	0,2-0,8	0,016-0,2	Diferencias admisibles para medidas sin indicacion de tolerancia	hasta 6	mas de 6 hasta 30	mas de 30 hasta 120	mas de 120 hasta 315	mas de 315 hasta 1000	mas de 1000 hasta 2000	mas de 2000 hasta 4000	mas de 4000 hasta 8000
SIGNOS A EXTINGUIR	▽	▽▽	▽▽▽	▽▽▽▽	DIN 7168 Precision media	±0,1	±0,2	±0,3	±0,5	±0,8	±1,2	±2	±3

PLANO DE REFERENCIA	N° SUSTITUIDO	NUEVO N°
-	-	-
FECHA-NOMBRE	-	-

Las medidas [ ] seran expres. verificadas.
Las medidas ( ) no tienen importancia.
Sin indicacion las esquinas seran a 0,5mm R.
Sin indicacion las aristas seran 0,5mm a 45°.

Sin indicacion expresa roscas de calidad 6 (media)
Tuercas 6H y Tornillos 6g.

ERICO CRUZ LEMUS	DESCRIPCIÓN	ANEXO 2	
	CONJUNTO	8.2.6. Conjunto Caja Pantalla PLC B	
	MAQUINA	TFC UVIC	
DIBUJADO 29/01/2010 E. Cruz	Hoja 1 de 1	NUMERO DE ARTICULO	ESCALA 1:5
VERIFICADO			Nº PLANO e000180
			ORDEN MOD. 0