UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño de un sistema automatizado para la medición e identiftcación de pistones de motor gasolina de Toyota del año 1970 al 2011 por medio de visión artiftcial y láser

Trabajo de graduación presentado por Juan José Tzun Monterroso para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2019

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

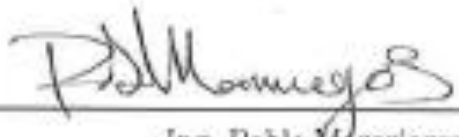
Diseño de un sistema automatizado para la medición e identiftcación de pistones de motor gasolina de Toyota del año 1970 al 2011 por medio de visión artiftcial y láser

Trabajo de graduación presentado por Juan José Tzun Monterroso para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2019

Vo.Bo.:

(f) 

Ing. Pablo Mazariegos

Tribunal Examinador:

(f) 

Ing. Pablo Mazariegos

(f) 

MSc. Carlos Esquit

(f) 

Ing. Otto Girón

Fecha de aprobación: Guatemala, 19 de Junio de 2019

Prefacio

A Dios, quien ha sido la piedra angular en todo este proceso.

A mi papá, por su esfuerzo diario para que yo lograra culminar mis estudios universitarios.

A mi mamá, por su apoyo incondicional.

A Lily Álvarez, por creer en mí y motivarme a ser mejor cada día.

Índice

Lista de figuras

Lista de cuadros

Resumen

El proyecto consiste en el diseño de un dispositivo electrónico capaz de tomar mediciones de los diámetros y las ranuras de los anillos de distintos tipos de pistones de un motor de gasolina. Por medio de un micrómetro digital Mitutoyo, se toman las medidas de los diámetros acercando el palpador a la superficie externa del pistón, para después enviar la lectura a un microcontrolador a través de comunicación serial. Para la medición de las ranuras se proyecta la sombra en un sensor CMOS de imagen lineal de 128 pixeles. La medida se calcula en base a la cantidad de pixeles que ocupe la sombra proyectada.

Además, con la ayuda del procesamiento de imágenes, se identifica la serie del motor al cual pertenece el pistón de muestra. Utilizando una cámara web se toma una imagen de la parte frontal y de la base del pistón, luego se comparan los rasgos más significativos con otros pistones que se encuentran en una base de datos. Con la medición precisa del diámetro y la identificación del pistón, se procede a desplegar toda su información mecánica que se encuentra en el catálogo Teikin, tal como: diámetro estándar, ancho de los anillos, cilindrada, tratamiento de superficie, diámetro del pasador, altura total, número de producto en inventario.

Por último, se evalúa la superficie del pistón que está en contacto con los cilindros del motor, en busca de rayaduras debido al desgaste por suciedad. Se utiliza un microscopio digital para la toma de imágenes, conectado por medio de USB a la computadora. Las imágenes se procesan a través de un algoritmo y luego se despliegan las imágenes de las zonas más afectadas.

Para el primer prototipo, se diseñó una estructura de barras lisas y roscadas para el desplazamiento vertical del pistón, el desplazamiento horizontal del micrómetro, un mecanismo de doble mandíbula para centrar y agarrar el pistón en una determinada posición. Todas las piezas se diseñaron en el software Autodesk Inventor 2018, para después realizarse en una impresora 3D en el departamento de electrónica.

Abstract

The project is about the design of an electronic device capable of measure the piston diameters and the rings grooves of different type of gasoline engine motor. By means of a Mitutoyo digital micrometer, the measurements of the diameters are taken bringing the measuring rod close to the external surface of the piston, then the reading is sent to a microcontroller through serial communication. The shadow of the grooves are projected into a CMOS image linear sensor, that has 128 pixels, and the measurement is calculated based on the number of pixels occupied by the shadow.

Additionally, the engine series to which the piston belongs is identified with the images processing. Using a web camera, a picture is taken of the front and the base of the piston, then the most significant features are compared to other pistons that are in a database. With the precise measurement of the diameter and identification of the piston, it proceeds to display all its mechanical information found in the Teikin catalog, such as: standard diameter, rings width, cylinders, surface treatment, piston pin diameter, total length, reference number.

Finally, the surface of the piston that is in contact with the cylinders of the engine is evaluated looking for scratches due to wear. A digital microscope is used for taking pictures, connected to the computer with USB connection. The images are processed through an algorithm and then the images of the most affected areas are displayed.

For the first prototype, a structure of smooth and threaded rods was designed for the vertical displacement of the piston, the horizontal displacement of the micrometer, a double jaw mechanism to center and grasp the piston in a certain position. All the pieces were designed in the software Autodesk Inventor 2018, and then made in a 3D printer in the electronics department.

CAPÍTULO 1

Introducción

Actualmente, el vehículo se ha convertido en una máquina compleja que forma parte de nuestra vida cotidiana. Por otra parte, el número de personas involucradas en la industria automovilística representa una cifra significativa, más que todo en brindar servicios de reparación, mantenimiento de vehículos, y venta de autopartes y repuestos. Tan solo en Guatemala se encuentran aproximadamente 3,000 establecimientos que brindan estos servicios, según un análisis realizado por el área de Inteligencia Comercial de Central America Data [1].

La reparación del motor de un vehículo consiste en efectuar mediciones, desmontar y desarmar, sustituir o mecanizar y montar nuevamente. Una parte importante en el trabajo de reparación es decidir si una pieza determinada debe cambiarse o conviene repararla. Esta decisión se toma en base a los años de experiencia del mecánico y con la ayuda de herramientas de medición especializadas.

En el presente proyecto se ha diseñado un dispositivo electrónico para automatizar el proceso que conlleva efectuar mediciones de una pieza en particular de todo el motor, que es el pistón. Con la ayuda de este dispositivo se toman la medida del diámetro, el ancho de las ranuras donde se colocan los anillos, se identifica la serie del motor al que pertenece y se identifican las zonas de la superficie con mayor desgaste en la falda del pistón debido a la suciedad.

CAPÍTULO 2

Antecedentes

2.1. Dispositivos electrónicos en el mercado

Actualmente, existen dispositivos electrónicos en el mercado que permiten medir el diámetro externo de los pistones con precisión y de forma automatizada, a diferencia de los micrómetros mecánicos convencionales, los cuales necesitan mayor manipulación y conocimientos por parte del usuario. Los hay tanto de contacto como sin contacto.

Los de contacto poseen una varilla en cada extremo que se acercan a la superficie del pistón, y dependiendo de la distancia que se contraigan, gracias a un mecanismo de resorte interno, se calcula el diámetro y se muestra al usuario en una pequeña pantalla [2] [3].

Por otro parte, se encuentran los de sin contacto. Estos dispositivos proyectan una cortina de luz láser hacia un sensor óptico. Al colocarse el pistón en la cortina, se crea una sombra que es percibido por el sensor. Por lo que el diámetro se calcula dependiendo de los pixeles que abarque la sombra en el sensor óptico [4]. Este principio también es utilizado para inspeccionar las ranuras donde se colocan los anillos, las tres caras de la superficie de la ranura se evalúan según los estándares de manufactura y mantenimiento [5] [6] [7].

2.2. Librería OpenCV

Dentro de la industria, se han adaptado los algoritmos de detección de objetos para la clasificación de los productos manufacturados en contenedores, o bien para que sean sometidos a otros procesos de manufactura. Una de las bibliotecas de visión de computador más utilizadas y potentes es OpenCV. Esta biblioteca cuenta con más de 2500 algoritmos optimizados [8]. Para esta aplicación en particular, se ha adaptado la herramienta para extraer las características más significativas de los pistones de cada serie de motor de gasolina de la marca Toyota.

2.3. Clasiftcadora de varillas con Mitutoyo y Arduino

Este proyecto consistió en crear una pequeña máquina que separa las varillas defectuosas de las que si cumplen los estándares de calidad. Se utilizan dos calibradores digitales Mitutoyo, que se encuentran en cada extremo de la varilla y se acercan a su superficie para medir la mitad del espesor cada uno. Luego los datos son enviados por medio de comunicación serial a un Arduino que se encarga de procesar la información y calcula el diámetro total [9].

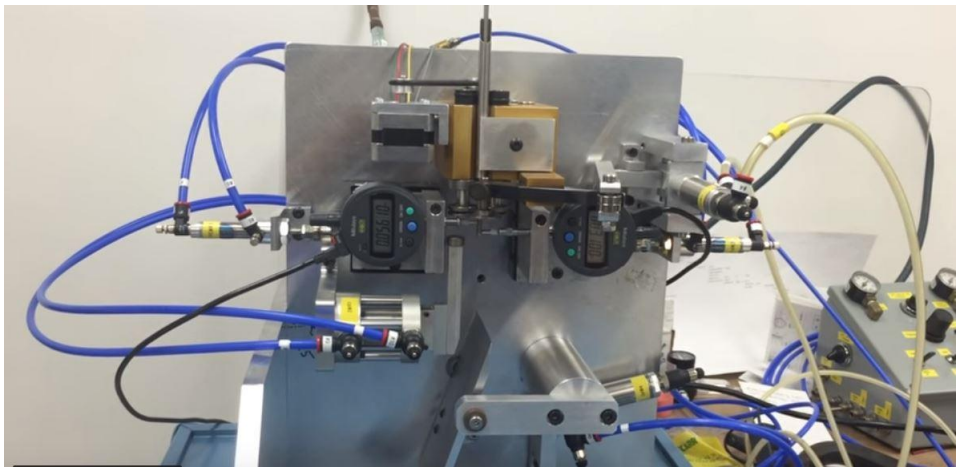


Figura 1: Medición de varillas con comparadores digitales Mitutoyo [9]

CAPÍTULO 3

Justificación

En el campo de la mecánica automotriz, se requiere que las personas involucradas en la inspección de las piezas mecánicas (cilindros, culata, eje cigüeñal, bielas, pistones, entre otros) estén capacitados para manipular las distintas herramientas de medición. Sin embargo, para las personas que no tienen la experiencia suficiente con los distintos tipos de motores que existen, se les dificulta realizar este proceso de inspección y les toma más tiempo al tomar la decisión de sustituir la pieza o someterla a un proceso de reparación [10].

Esta inseguridad puede provocar la toma de mediciones erróneas y, por consiguiente, una mala decisión que se ve reflejado en un coste total de reparación engañoso.

Una de las partes más esenciales del motor de un automóvil son el conjunto del eje cigüeñal, compuesto por las bielas y los pistones. Se encargan de transformar la energía de combustión en energía mecánica que mueven las ruedas.

Con el objetivo de automatizar el proceso de inspección de los pistones, se ha diseñado un dispositivo electrónico que facilite la toma de medidas correspondientes, además de guiar al personal a identificar la serie de motor al que pertenece. Se disminuye así el tiempo que conlleva efectuar la inspección, el margen de error en las mediciones, y se toma una mejor decisión al sustituir o no el pistón de muestra.

Este trabajo es de importancia debido a que estoy afianzando, no solo diversos conocimientos adquiridos a lo largo de mi carrera universitaria, sino también parte de los años de experiencia que tiene el negocio familiar en reparación e inspección del motor de un automóvil.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Diseñar un dispositivo electrónico para la medición del diámetro y ancho de las ranuras de los anillos de un pistón e implementar un algoritmo de visión por computador para el reconocimiento de los pistones de motor gasolina de vehículos Toyota del año 1970 al 2011.

4.2. Objetivos específtcos

- Implementar y diseñar un mecanismo para la medición del diámetro de un pistón con un comparador digital.
- Implementar un sensor de imagen lineal CCD para medir el ancho de las ranuras de los anillos de un pistón.
- Implementar una cámara de exploración de línea para la detección y medición de las ranuras de los anillos del pistón.
- Medir el desgaste que posee la ranura del anillo de un pistón, para determinar si existe efecto de aleteo que pueda reducir la efectividad de los anillos.
- Identificar si existen áreas con desgaste debido a la suciedad en la falda del pistón a través de un microscopio digital, para decidir sustituirlo por otro nuevo.
- Utilizar el procesamiento de imágenes para el reconocimiento de cada tipo de pistón, según la serie de motor al que pertenece.
- Diseñar y fabricar un mecanismo capaz de posicionar fijamente un pistón para que permita efectuar las mediciones del diámetro y toma de captura de imágenes.

- Crear una base de datos de imágenes sobre pistones de motor gasolina de vehículos Toyota del año 1970 al 2011 para complementar el procesamiento de imágenes, usando como referencia el catálogo de pistones Teikin.
- Crear una interfaz gráfica amigable al usuario que le permita controlar remotamente el dispositivo y visualizar los resultados obtenidos del procesamiento de imágenes y de las mediciones.

CAPÍTULO 5

Alcance

El presente trabajo de graduación tiene como propósito el diseño de un dispositivo electrónico capaz de inspeccionar los pistones usados de un motor de automóvil tomando las medidas del diámetro, la medida del ancho de las ranuras de los anillos, identificar la serie del motor al que pertenece y encontrar áreas de mayor desgaste en la superficie.

Este diseño busca ser una herramienta que pueda guiar al personal de una reconstructora de motores, involucrado en el proceso de mantenimiento correctivo del motor, al momento de dar un diagnóstico general del estado actual de los pistones.

Al ser un prototipo, no se pretende que sea más preciso que los que existen actualmente en el mercado, pero se busca abarcar los aspectos más importantes que puedan definir si un pistón usado se pueda volver a usar o si es necesario sustituirlo.

Además, el presente prototipo solo puede inspeccionar pistones de la marca Toyota y de un motor de gasolina. La base de datos del sistema solo posee información de esta rama de automóviles, haciendo énfasis también que el modelo más reciente tabulado data del año 2011.

CAPÍTULO 6

Marco teórico

6.1. Motor de combustión interna

Se le denomina así al motor que convierte la energía térmica en energía mecánica a través de la combustión de aire y carburante que se quema interiormente [11]. El trabajo se obtiene mediante el desplazamiento lineal del émbolo de un mecanismo biela-manivela [12].

Los motores de combustión interna usados en los automóviles deben reunir una serie de cualidades, entre ellos:

- Buen rendimiento
- Bajo consumo en relación a su potencia
- Gases de escape, pocos contaminantes
- Fiabilidad y durabilidad
- Bajo coste de fabrición y mantenimiento

Los vehículos autopropulsados que utilizan este tipo de motor son: automóviles y motocicletas; camiones y autobuses; maquinaria agrícola y de obras públicas; locomotoras, barcos y aviones ligeros

6.1.1. Clasiftcación de los motores de combustión interna

Los motores de los automóviles están clasificados de diferentes formas, aunque van variando según el fabricante y el diseño, los tipos básicos siguen siendo los mismos.

Cuadro 1: Clasificación de los MCI [11]

Por el inicio de la combustión	Por el ciclo de trabajo	Por el movimiento del pistón
Motor de encendido por chispa (Otto)	Motor de 4tiempos	Motor de pisón alternativo
Motor de encendido por compresión (diésel)	Motor de 2tiempos	Motor rotativo

Motor Otto: También llamado motor de explosión o motor de encendido provocado. Consume una mezcla de aire y gasolina, la mezcla se inflama por una chispa eléctrica proporcionada por un sistema de encendido externo. Soporta presiones moderadas, por lo que sus componentes son ligeros y pueden alcanzar revoluciones altas. Consiguen su potencia máxima entre 5000 y 7000 revoluciones por minuto [11].

Motor de pistón alternativo: Este motor transmite el trabajo mediante pistones, éstos se desplazan con un movimiento lineal alternativo y lo transforman en movimiento de rotación gracias a un sistema de biela-manivela [11]. Número de cilindros usados comúnmente:

- Motocicletas -> de 1 a 4
- Automóviles -> de 2 a 6 en línea y de 6 a 8 en V
- Camiones -> de 4 a 6 en línea y de 8 a 12 en V

Motor de cuatro tiempos: Desarrolla su ciclo de trabajo en cuatro carreras del pistón. En cada carrera se realiza una fase de: admisión, compresión, expansión y escape. El ciclo se completa en dos vueltas de cigüeñal. La admisión de gases en el cilindro y la expulsión de los mismos ya quemados, se controlan mediante válvulas que abren y cierran los conductos de admisión y escape. El ciclo de cuatro tiempos se utiliza tanto en motores Otto como en motores diésel [11].

6.1.2. Constitución del motor de combustión interna de cuatro tiempos

La estructura básica del motor está formada por el bloque y la culata, sobre estas piezas se montan todos los demás elementos que lo conforman.

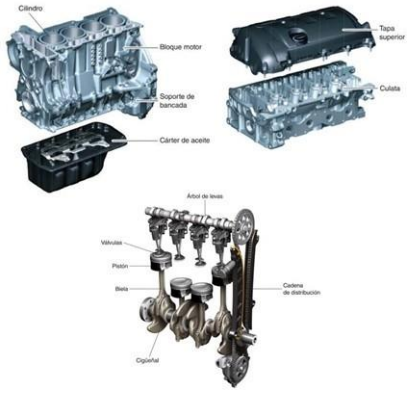


Figura 2: Constitución de un motor de combustión [11]

El pistón

Es el émbolo del mecanismo biela-manivela y aumenta o disminuye el volumen del cilindro. En el pistón pueden diferenciarse dos partes: la cabeza y la falda. La cabeza del pistón forma parte de la cámara de combustión y está expuesta a temperaturas más altas que la falda. Esto se debe a que no puede ser enfriado directamente por el refrigerante o el aire del exterior. Debido a esto, el diámetro de la cabeza se ha fabricado más pequeño que el diámetro de la falda, considerando también la expansión resultante por la diferencia de temperaturas. Adicionalmente, el diámetro en la dirección del pasador se ha fabricado más pequeño que el diámetro en la dirección perpendicular a éste. Como resultado de estas variaciones en los diámetros, el pistón tiene una forma oval en la vista superior, y forma de cono truncado circular en la vista lateral [13].

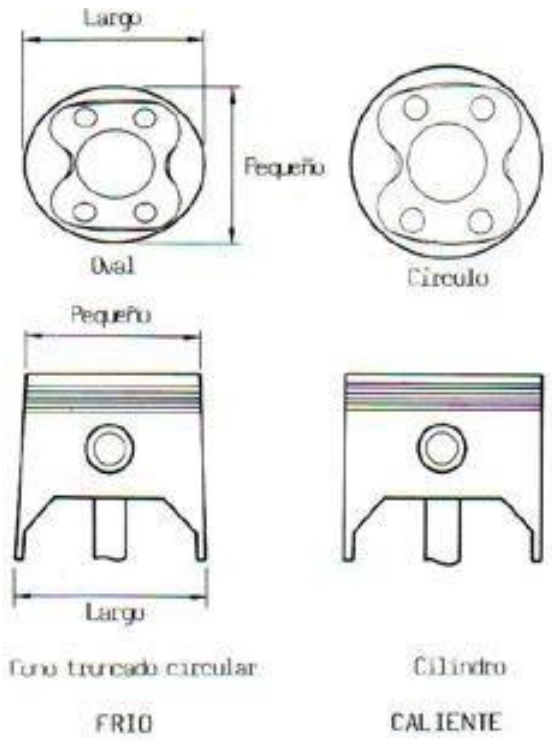


Figura 3: Forma del pistón en frío y caliente [13]

Si la holgura de uno o todos los pistones excede el límite de servicio debido al desgaste del cilindro o de los pistones, ya sea el bloque de cilindros o los pistones deben ser reemplazados, o ya sea los cilindros deben ser rectificados y deben usar pistones en sobre medida.

Cuando un cilindro es rectificado, el tamaño del pistón en sobre medida es determinado mediante la magnitud del desgaste del cilindro. Por lo regular, pistones en sobre medida de 0.5 están disponibles como piezas de repuestos para la mayoría de motores. También pueden encontrarse pistones en 0.25 o 0.75 para ciertos motores.

Al decir que el diámetro exterior de un pistón es 0.50 se dice que es aproximadamente 0.50 mm más grande que el diámetro de un pistón de tamaño estándar [13].

Debido a la forma no cilíndrica del pistón, la medida del diámetro debe tomarse en la

zona adecuada que viene incluida en los catálogos, generalmente a una distancia x de la cabeza. Generalmente, los pistones se miden con un micrómetro a temperatura ambiente.

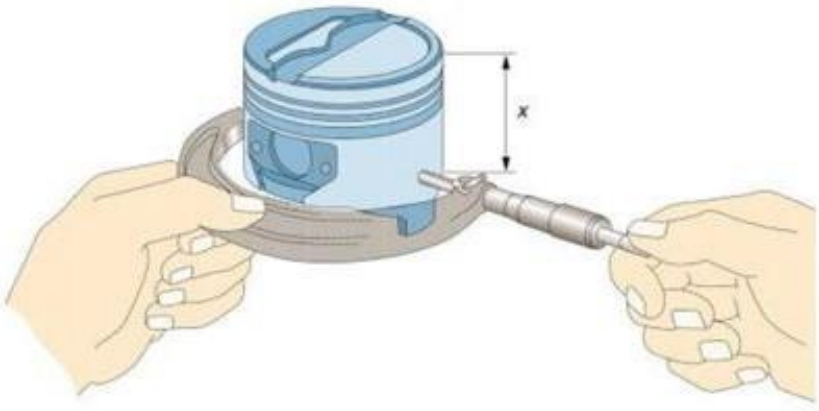


Figura 4: Medición del diámetro de un pistón [11]

Figura 5: Partes de un pistón

6.2. Útiles de medida en mecánica automotriz

Son todos aquellos instrumentos diseñados y adaptados para comparar o medir las magnitudes que se utilizan en el automóvil y en las reparaciones del mismo. El útil emplea una unidad de medida establecida por el fabricante (milímetros, bar, etc) y permite realizar las mediciones de forma directa o indirecta. Estos instrumentos incorporan un dispositivo visualizador o indicador [14].

6.2.1. Micrómetro o palmer

Es un instrumento de medida directa que permite realizar medidas de hasta una milésima de milímetro, es decir 0.001 mm. Los más empleados tienen una menor variación de medida de una centésima, o sea 0.01 mm. El principio de funcionamiento de un micrómetro está en el avance de un tornillo sobre una tuerca fija. Si se tiene, por ejemplo, un paso de tornillo de 0.5 mm por cada vuelta, al dividir esa vuelta en 50 divisiones se tiene un valor de 0.01 mm por cada división [14]. Puede verse en la siguiente ecuación:

$$Apreciacion = \frac{1\ mm}{50\ divisiones\ x\ 2\ vueltas} = \frac{1}{100} = 0.01\ mm \tag{1}$$

Para realizar una medición en un micrómetro, se debe colocar la pieza que se desea medir, entre las superficies de contacto y girando el tambor por medio del trinquete. El trinquete es un embrague que determina una presión adecuada sobre la pieza, evitando el apriete del tornillo y el falseo de la medición [14].

Figura 6: Partes del micrómetro de exteriores [14]

6.2.2. Reloj comparador

Este es el instrumento de medición más empleado e importante en los instrumentos de comparación. Para obtener mediciones precisas con este instrumento, lo primero es fijar el reloj en un soporte adecuado, un soporte que esté diseñado para realizar la medición. Luego, se acerca el palpador a la pieza que se desea medir. La escala del reloj está dividida en 100 partes, por lo que tiene una apreciación de centésimas de milímetro. Los desplazamientos del palpador se reflejan en las mediciones de la escala principal mediante un mecanismo de precisión.

Figura 7: Componentes del reloj comparador analógico [14]

Comparador digital Mitutoyo

Su funcionamiento es similar al analógico, con la diferencia de incorporar tecnología digital. Esta marca en especial permite la transferencia de datos a través de una salida SPC. Las especificaciones técnicas de la serie 543-783B se detallan a continuación.

Cuadro 2: Especificaciones técnicas del comparador digital Serie 543-783B [15]

Resolución	0.01 mm
Exactitud	0.0008 in
Interfaz	LCD
Rango de medición	12.7 mm
Nivel de protección contra agua y polvo	IP42
Batería	SR44

Figura 8: Comparador digital Mitutoyo 543-783B [15]

6.3. Sensores de imágenes o visión

Son dispositivos de estado sólido que convierten una imagen óptica en una señal analógica línea por línea. Existen dos tipos de sensores con configuraciones de circuitos diferentes: los sensores de imagen CMOS y los sensores de imagen CCD. Son adecuados para aplicaciones como escáner de fotocopiadoras, lectores de código de barras, cámara de escaneado lineal para exámenes visuales, clasificador de colores de grano, entre otros [16].

Están fabricados con materiales semiconductores, y están estructurados en forma de una matriz. Funcionan al acumular una carga eléctrica en cada celda (llamado pixel) de esa matriz proporcionalmente a la intensidad de la luz que incide sobre ella. A mayor intensidad luminosa, mayor carga se acumula [17].

Además, estos sensores cuentan con un protocolo específico de comunicación, como SPI, Serial USB, entre otros. Esto depende de la marca y el modelo del sensor de imagen [18].

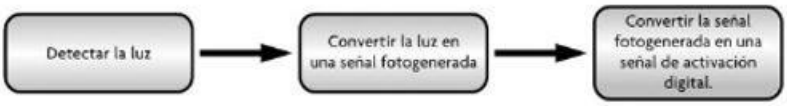


Figura 9: Tareas realizadas por un sensor de visión [18]

6.3.1. Cámara de escaneo de línea

Este tipo de cámaras utilizan un sensor unidimensional que adquieren una imagen unidimensional en un solo cuadro. Se usan en aplicaciones donde se requiere inspeccionar objetos en movimiento. El principio de funcionamiento de este tipo de cámaras es similar a un escáner de documentos [19].

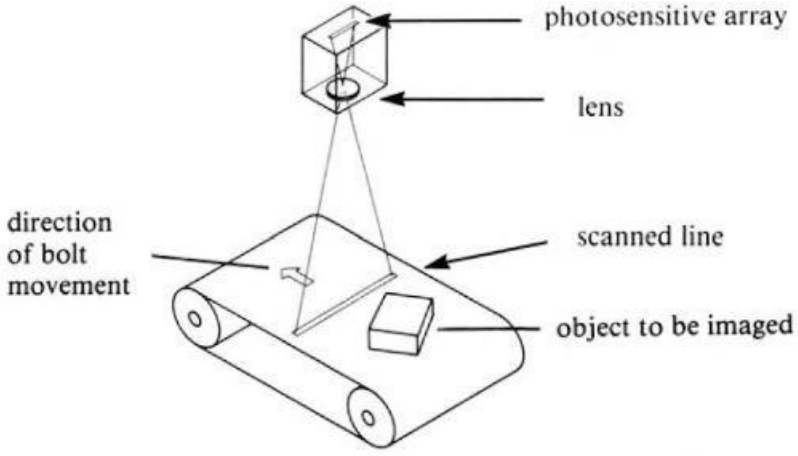


Figura 10: Aplicación de una cámara de escaneo de línea en una banda transportadora [19]

Estas cámaras necesitan de buena iluminación para realzar los rasgos de interés de un objeto. Típicamente se usan luces de línea, que es luz concentrada de una línea de LEDs,

para iluminar el área del objeto que está siendo observado. Usualmente, las luces de línea se posicionan atrás de los objetos para dar un efecto de iluminación desde el fondo, o también desde un ángulo alto como mirando hacia abajo en el objeto. La iluminación desde el fondo es utilizado para detectar agujetos pequeños en materiales opacos o partículas defectuosas en materiales transparentes. La iluminación con ángulo es usado para obtener los contornos o detectar los defectos en la superficie.

$$Size = \frac{3 \cdot FOV}{minimum\,defect\,size} \tag{2}$$

Para determinar la cámara que se necesita para determinada aplicación es necesario tener en cuenta la sensibilidad, el tamaño en pixeles y la velocidad de línea. Para determinar el tamaño es necesario especificar el campo de visión (FOV) y el tamaño mínimo de defecto. La cámara debe tener la suficiente resolución como para tener al menos 3 o 4 pixeles que cubran el tamaño mínimo de defecto [20].

Módulo de cámara TSL1401CL para Arduino

Este módulo se caracteriza por utilizar un sensor TSL1401CL del tipo CCD con un arreglo de 128 pixeles. Permite modificar el valor de resistencia para mejorar la señal en la salida. Sus especificaciones técnicas se detallan a continuación.

Cuadro 3: Especificaciones técnicas módulo de cámara TSL1401CL

Voltaje de 3V a 5V
Tamaño de 29mmx29mm
Peso 10 g
Recuento de pixeles de alta resolución
Lente ultra angular
Distancia focal 10mm a 18mm

6.4. Visión por computadora

Es un proceso que consiste en la extracción de información del mundo real a través de imágenes, utilizando como herramienta principal un computador. Reúne las técnicas necesarias para el procesamiento de imágenes y simular las tareas que el sistema de visión humano es capaz de realizar [21].

Cuando se habla de procesamiento de imágenes, se hace referencia al proceso de tomar una imagen y producir una versión modificada de esa imagen. Tiene que ver con la adquisición, transmisión, procesamiento y representación de las imágenes. Son todas las técnicas que se utilizan para mejorar la apariencia visual de las imágenes para un observador y para preparar, de forma conveniente, el contenido fotográfico a la percepción por parte de las máquinas [22].

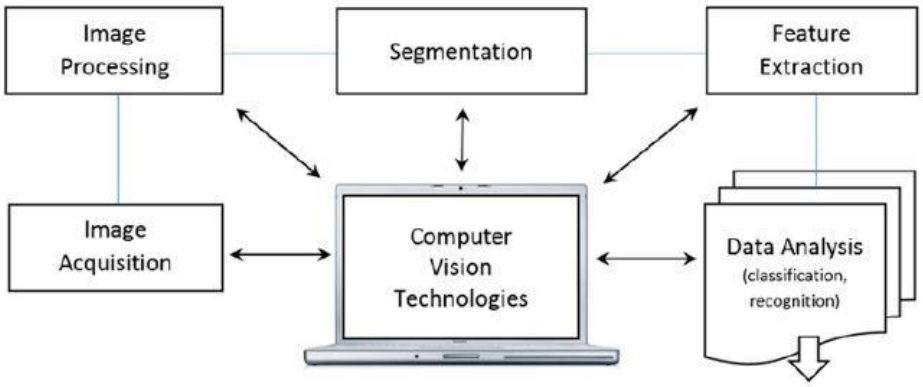


Figura 11: Diagrama de la visión por computador [23]

6.4.1. Etapas de un proceso de visión artiftcial

Las etapas pueden clasificarse en dos grupos. En el primer grupo se encuentran las etapas que ejecutan métodos de bajo nivel, que sirven para obtener las características más básicas de la imagen, tales como: bordes, regiones y otros atributos simples. Por otro lado, están las etapas que realizan un procesamiento de la imagen de alto nivel, donde se recogen las características extraídas del bajo nivel y se construye una descripción de lo que se va a analizar [21].

Figura 12: Etapas del proceso de visión por computador [21]

Se describe de forma breve qué es cada una de estas etapas:

- Adquisición de la imagen: Se captura una proyección 2D del objeto mediante la proyección de la luz en el espacio de trabajo.
- Reprocesamiento: Se elimina todo tipo de ruido, se recorta la imagen y se rectifica.
- Segmentación: Aplicación de diferentes filtros para aislar los objetos deseados.
- Extracción de características: Se obtienen las características de interés del objeto para luego analizarlas.

- Reconocimiento y localización: Se reconoce el objeto que se encuentra en la escena a través de las características previamente extraídas.
- Interpretación: Con la información obtenida se hace una interpretación de lo que se tiene en la escena.

6.5. OpenCV

Es una librería multiplataforma y de código abierto que brinda funciones para la visión por computador. Brinda interfaces de alto nivel para la captura, procesamiento y presentación de la información de la imagen [8]. La biblioteca está escrita en C y C++, puede ejecutarse en Windows, Linux y Mac OS X. Es un desarrollo activo en las interfaces para Ruby, Python, Matlab y otros lenguajes.

Uno de los principales objetivos de la creación de ésta librería fue hacer de la visión por computador una infraestructura fácil de usar, que ayude a las personas a desarrollar aplicaciones de visión sofisticadas y de una forma más rápida. Contiene más de 500 funciones que cubren muchas áreas en visión, tales como: seguridad, inspección de productos de fábrica, imágenes médicas, visión estéreo y robótica.

6.5.1. Thresholding

Esta función permite clasificar los pixeles de una imagen de acuerdo a su valor de color en escala de grises, eliminando aquellos que no cumplan con el valor predefinido, llamado umbral. Se crea una imagen binaria que incluye los pixeles que superan el valor del umbral y los que no. [24]

También se tiene la función *adaptive threshold* el cual permite variar el valor del umbral dependiendo del método adaptativo aplicado a la imagen. Si el método se configura como CV_ADAPTIVE_THRESH_MEAN_C, todos los pixeles del área son ponderados por igual. Pero si se elige CV_ADAPTIVE_THRESH_GAUSSIAN_C, entonces todos los pixeles de la región cerca de (x,y) son ponderados de acuerdo a una función Gaussiana de su distancia desde el punto central de la imagen [25]. Cuando hay mucha iluminación o gradientes reflejantes que se desean umbralizar dependiendo del gradiente de intensidad de la imagen.

6.5.2. Detección de bordes

Consiste en la detección de los bordes más marcados en la imagen de estudio y como resultado se obtiene una imagen binaria. Lo que se hace es dibujar líneas blancas en un fondo negro para indicar los bordes que se encontraron, todo lo demás se descarta. Esta función se puede describir como una operación de filtrado de paso alto [26]. Los filtros empleados en la detección de bordes son:

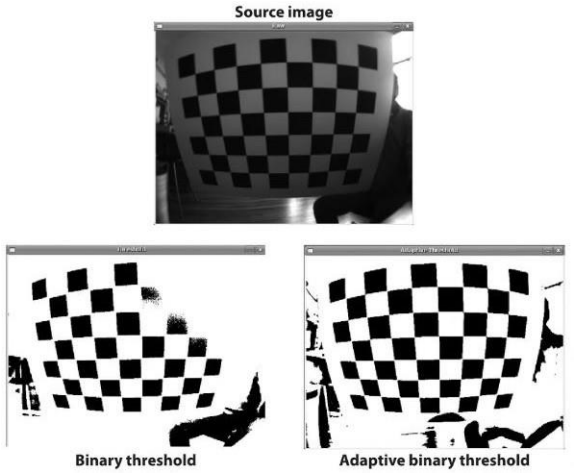


Figura 13: Aplicación de threshold y threshold adaptativo en una imagen [25]

Sobel

Solamente permite la detección de bordes horizontales o verticales, dependiendo del kernel (3) que se elija. El kernel de la derecha encuentra los bordes verticales, mientras que el de la izquierda los bordes horizontales.

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{3}$$

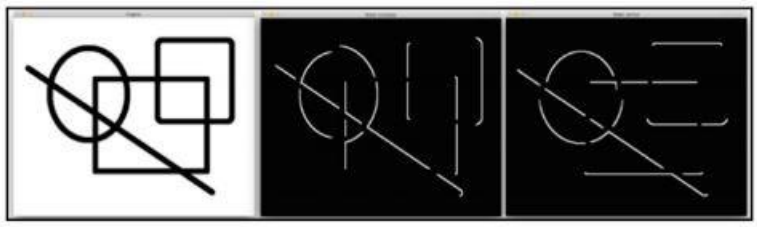


Figura 14: Aplicación filtro sobel vertical y horizontal [26]

Laplaciano

A diferencia del anterior, permite obtener una imagen con los bordes tanto horizontales como verticales. No siempre funciona correctamente porque da lugar a mucho ruido en la salida.

Canny

Fue creado por J. Canny en 1986 [25]. Esta función es la que mejor se adapta a imágenes más complejas, no produce tanto ruido como el laplaciano, obteniendo así una mejor calidad

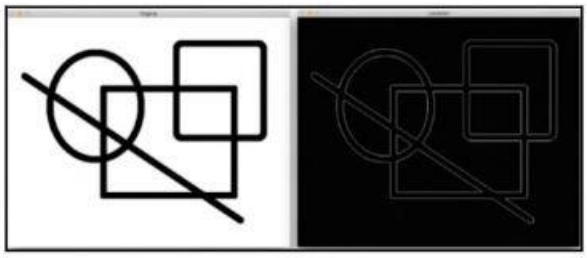


Figura 15: Aplicación de filtro laplaciano [26]

que los dos anteriores. Este filtro va detectando los bordes dependiendo de los dos valores de umbral que tiene como parámetros, mientras más altos sean estos umbrales los bordes menos significativos serán ignorados. Estos valores se conocen como umbral alto y umbral bajo [26].

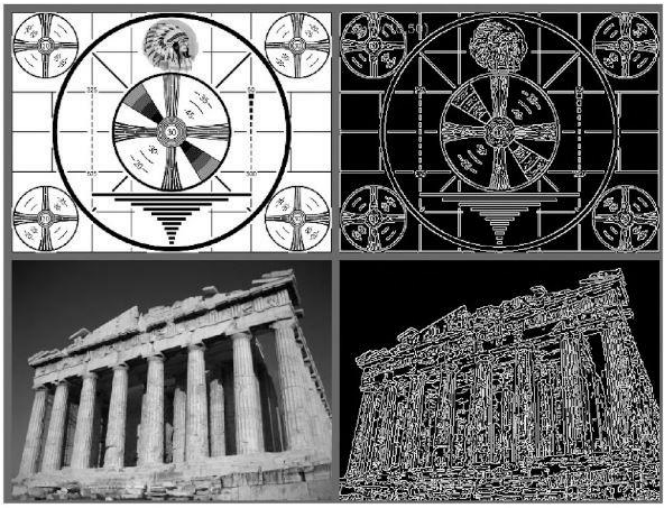


Figura 16: Detección de bordes Canny para dos imágenes con un umbral alto de 50 y umbral bajo de 10 [25]

6.5.3. Image Hashing con Python

Es el proceso de examinar el contenido de una imagen y construir un valor hash que es un valor numérico que representa a la imagen basándose en su apariencia visual. Para imágenes que son parecidas deberían de tener un hash similar. Las características más relevantes en una imagen son usadas para generar una huella digital distinta, pero no única, y pueden ser comparadas con otras [27]. En Python se encuentra una librería llamada ImageHash que reúne cuatro algoritmos que calculan el hash de una imagen aplicando diferentes técnicas:

- Average hashing (aHash): Convierte la imagen en una imagen de 8x8 en escala de grises y crea el hash en función de si el valor del pixel es mayor que el color promedio de la imagen.

- perception hashing (pHash): Similar al aHash con la diferencia en el uso de una transformada de coseno discreta y compara los pixeles en base a frecuencias.
- Difference hashing (dHash): A diferencia de los anteriores, rastrea los gradientes, es más rápido y fácil de implementar y es mucho más preciso. Calcula la diferencia de cada pixel y lo compara con el promedio de todas las diferencias.
- Wavelet hashing (wHash): Trabaja en el dominio de la frecuencia como lo hace pHash pero usa una transformación discreta de la onda en vez de coseno.

Los primeros tres algoritmos fueron creados en base a los pasos que plantea el Dr. Neal Krawetz en su blog [28]. El algoritmo wHash fue creado por Dmitry Patrelov [29]

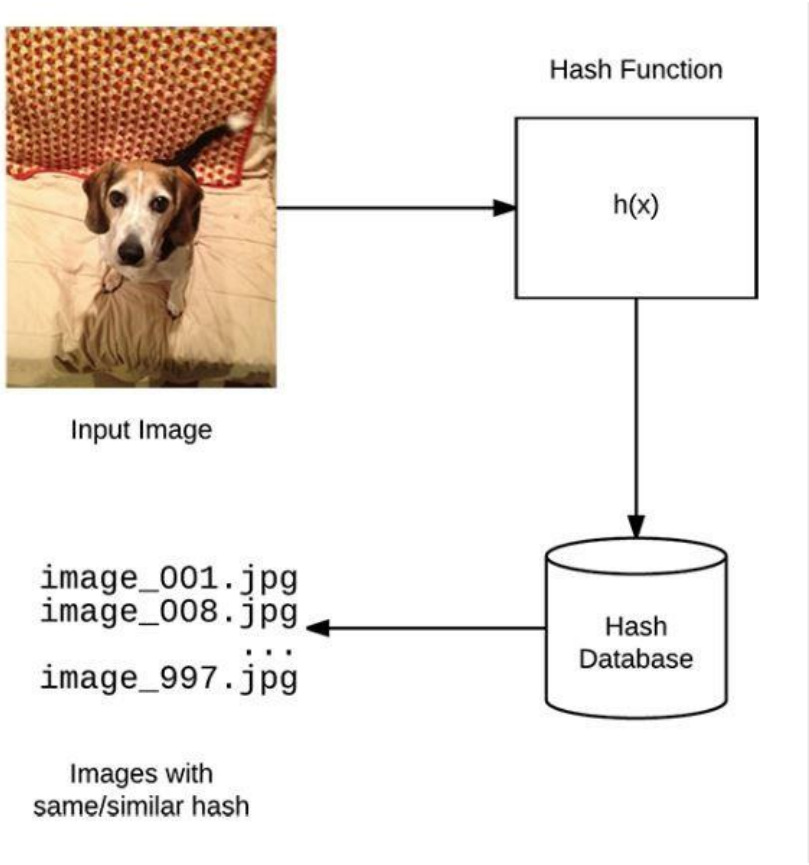


Figura 17: Diagrama del funcionamiento de Image Hashing [27]

6.6. Detección de objetos en visión de computador con MATLAB

La detección de objetos es el proceso de encontrar objetos del mundo real como caras, bicicletas y edificios en imágenes o videos. Los algoritmos de detección de objetos generalmente usa la extracción de características y algoritmos de aprendizaje para reconocer instancias de una categoría de objetos. Se usa comúnmente en aplicaciones como sistemas de recuperación de imágenes, seguridad, vigilancia y asistencia avanzada para el conductor [30].

6.6.1. Machine Learning

Es una ténica para el análisis de datos que enseña a los ordenadores a realizar determinado tipo de tareas que para las personas y los animales resulta natural, aprendiendo a través de la experiencia. Estos algoritmos de aprendizaje automático utilizan métodos de cálculo para aprender directamente de los datos sin depender de una ecuación. Los algoritmos tienen un mejor funcionamiento a medida en que aumenta el número de muestras disponibles para el aprendizaje. [31].

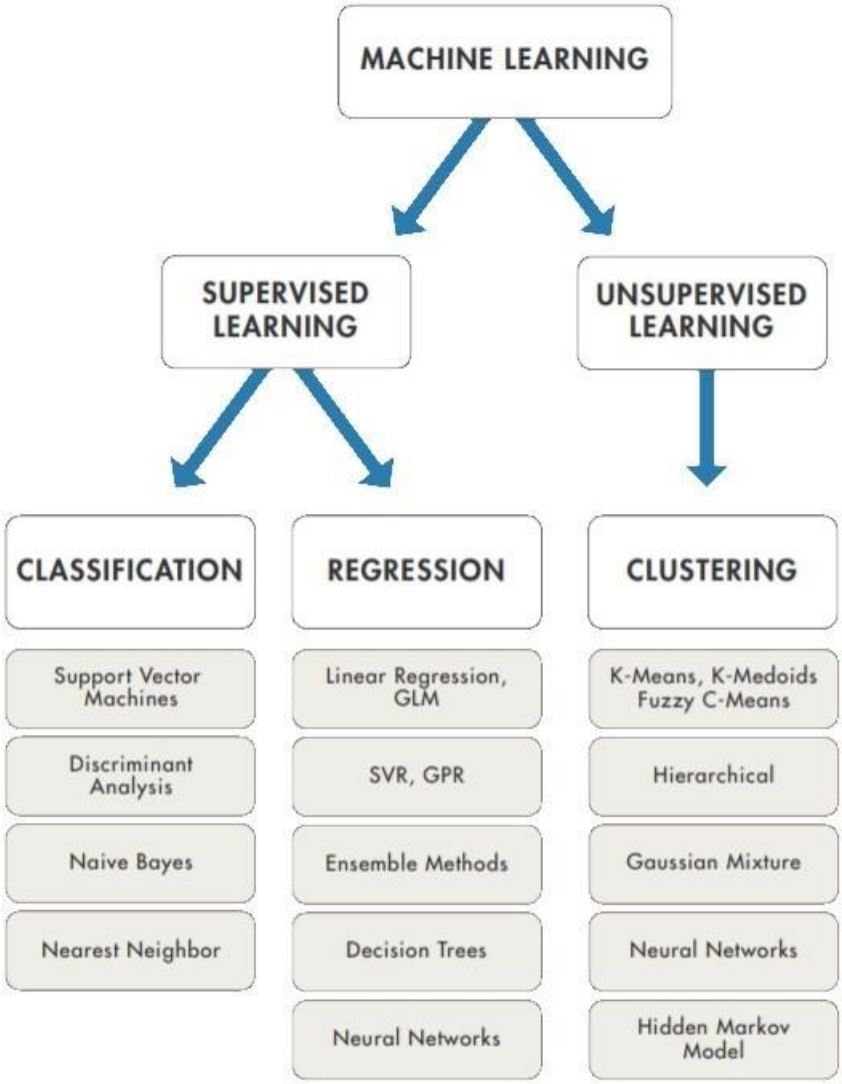


Figura 18: Técnicas de aprendizaje automático: aprendizaje supervisado y aprendizaje no supervisado [31]

Aprendizaje supervisado

El objetivo del aprendizaje supervisado es construir un modelo que pueda predecir en base a cierta incertidumbre. Se toma un conjunto de datos conocidos y se entrena un modelo que pueda generar predicciones como respuesta a nuevos datos. Este tipo de aprendizaje utiliza técnicas de clasificación y regresión para desarrollar modelos de predicción:

- Técnicas de clasificación: Predice respuestas discretas. Los modelos de clasificación son entrenados para clasificar datos en categorías. Entre sus aplicaciones están medicina, reconocimiento de voz, y puntuaciones de crédito.
- Técnicas de regresión: Predice respuestas continuas. Entre sus aplicaciones están predicción de precios en acciones, reconocimiento de escritura, y procesamiento de señales acústicas.

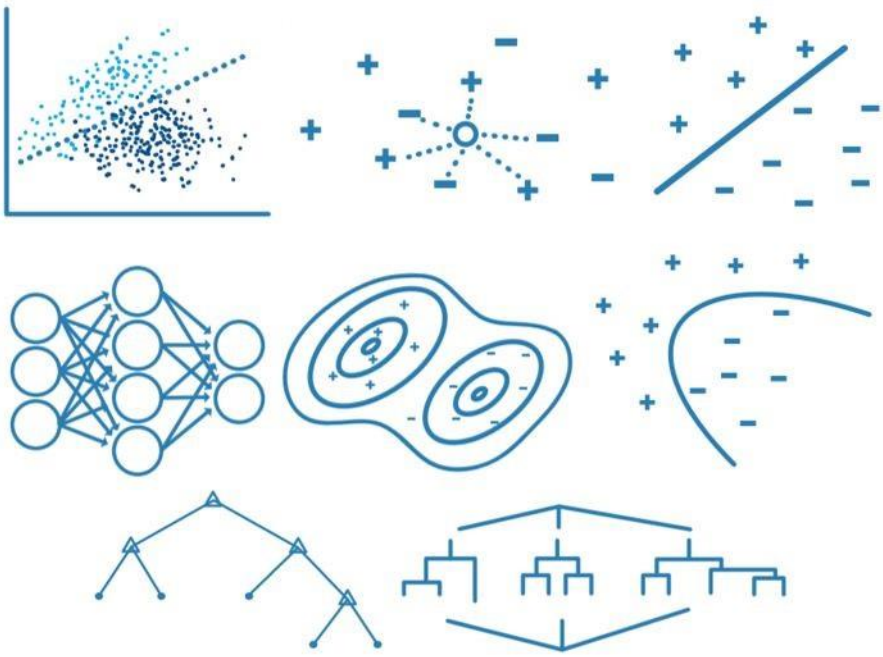


Figura 19: Algoritmos de aprendizaje supervisado

6.6.2. Detección y extracción de Features

Los Features locales y sus descriptores son los bloques de construcción de muchos algoritmos de visión por computador. Entre sus aplicaciones están el registro de imágenes, detección de objetos y clasificación, seguimiengo y estimación de movimiento. Estos algoritmos soportan mejor los cambios en el tamaño, rotación y oclusión. El toolbox de Matlab incluye los descriptores SURF, FREAK, BRISK, LBP, ORB y HOG [32].

La extracción de Features, es un tipo de reductor dimensional que representa de manera eficiente las partes de interés de una imagen como un vector compacto de características.

Es muy útil cuando las imágenes son grandes y se requiere una reducida representación de características para completar tareas como la comparación y recuperación de imágenes [33].

SURF Features

El SURF (Speed-up Robust Features) es un algoritmo de aprendizaje automático compuesto por cinco pasos: Cálculo de la matriz Hessian, generación del espacio de escala, selección de los feature points, selección de la dirección de los feature points y construcción del operador [34].

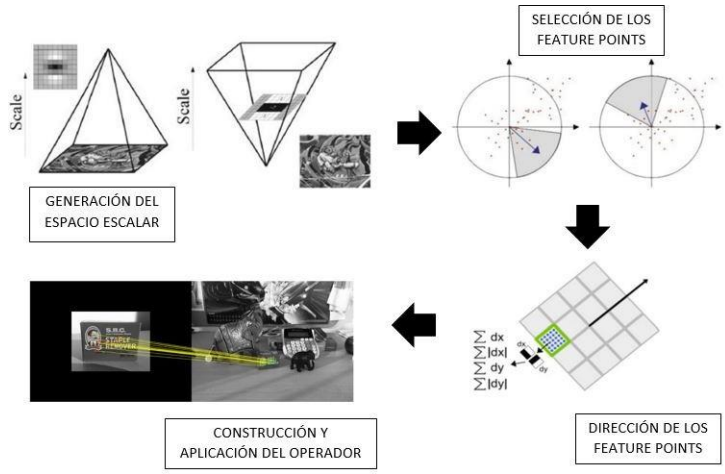


Figura 20: Procedimiento del algoritmo SURF y su aplicación

HOG Features

El HOG (Histogram of Oriented Gradients) es un descriptor de Features de una imagen. Primeramente calcula las orientaciones del gradiente de cada pixel, luego calcula un histograma por cada orientación en una región rectangular pequeña y finalmente se crea un vector concatenando los histogramas de cada región pequeña [35].

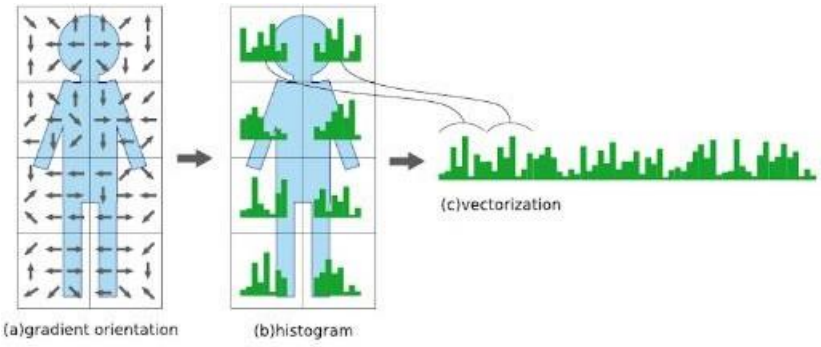


Figura 21: Cálculo del HOG de una imagen [35]

6.6.3. Classiftcation Learner App

Es una aplicación propia de Matlab que emplea el algoritmo de aprendizaje supervisado de Machine Learning, utilizando un conjunto de datos conocidos (datos de entrenamiento) para realizar predicciones. A partir de ello, el algoritmo busca crear un modelo que pueda realizar predicciones acerca de los valores de respuesta para un nuevo conjunto de datos. Frecuentemente se utiliza un conjunto de datos de prueba para validar el modelo. Este tipo de aprendizaje incluye dos categorías de algoritmos: clasificación y regresión [36].

Entre los algorimos de clasificación comunes se encuentran: máquinas de vectores de soporte (SVM), redes neuronales, clasificador Naive Bayes, árboles de decisión, análisis discriminante, vecinos más cercanos (kNN)

Entre los algorimos de regresión comunes se encuentran: regresión lineal, regresión no lineal, modelos lineales generalizados, árboles de decisión, redes neuronales

Para usar el modelo con nuevos datos, es posible exportar el modelo desde la aplicación al workspace o generar el código en MATLAB.

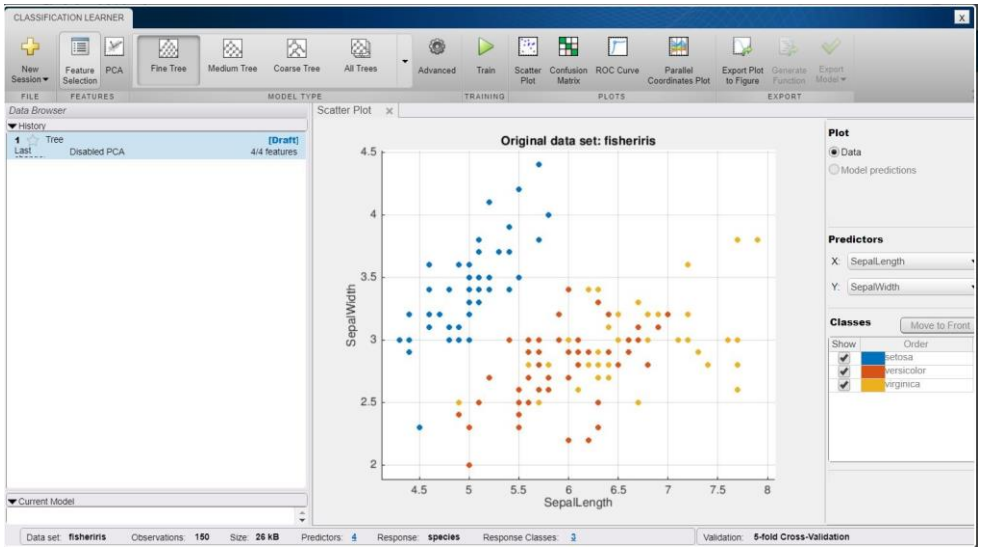


Figura 22: Ventana principal Classification Learner de MATLAB [36]

Máquinas de vectores de soporte (SVM)

Clasifican los datos encontrando el límite lineal de decisión (hiperplano) que separa todos los puntos de datos de una clase de otra. El mejor hiperplano para un SVM es aquel con mayor margen entre dos clases, cuando los datos son linealmente separables. Si los datos no son linealmente separables, una función de pérdida es usado para penalizar los puntos que estén en el lado equivocado del hiperplano.

6.7. La cámara digital

La cámara de un sistema de visión por computador es el dispositivo que recibe la luz reflejada por la escena y la utiliza para generar imágenes. El elemento más importante de la cámara es su sensor, formado por una capa de material fotosensible que transforma la luz incidente en señales eléctricas. La tecnología digital más utilizada actualmente para la construcción de sensores de imagen es CCD [21].

6.7.1. Factores importantes de una cámara digital

Píxeles

El pixel es una unidad básica de información de una imagen digital. Se representa mediante un punto o cuadrado en la pantalla de visualización de un monitor de computadora. Según la cantidad, el tamaño y la combinación de colores de los píxeles, se puede medir la resolución de la cámara.

Iluminación

La iluminación es la característica que utilizan las ténicas de visión por computador para extraer información de los objetos de una escena. La iluminación de una escena se puede realizar con luz natural o mediante la utilización de lámparas. Las tecnologías de iluminación más utilizadas son lámparas halógenas, lámparas fluorescentes, diodos LED y láser [21].

Óptica

Es uno de los elementos más importantes de cualquier sistema de visión. Su objetivo es concentrar la luz reflejada por los objetos de la escena en el sensor de la cámara para generar la imagen. Está formada por un conjunto de lentes positivas o convergentes y negativas o divergentes, que aplican una transformación a la luz recibidad de la escena, definida matemáticamente mediante una proyección perspectiva. Esta transformación consiste en una inversión y escalado de los objetos de la escena [21].

- Lente normal: cubre la película con un campo de visión que corresponde aproximadamente a la visión normal de una persona [37].
- Lente gran angular: tiene una longitud focal más corta que la normal, incluye un campo de visión más ancho y los objetos parecen más pequeños [37].
- Lente zoom: es aquella cuya longitud focal se puede modificar de modo que parece que el sujeto o el objeto cuya imagen se va a capturar, se acerque o se aleje conforme cambia la longitud focal [37].
- Lente macro: permiten realizar enfoques a muy corta distancia del sujeto u objeto a fotografiar.

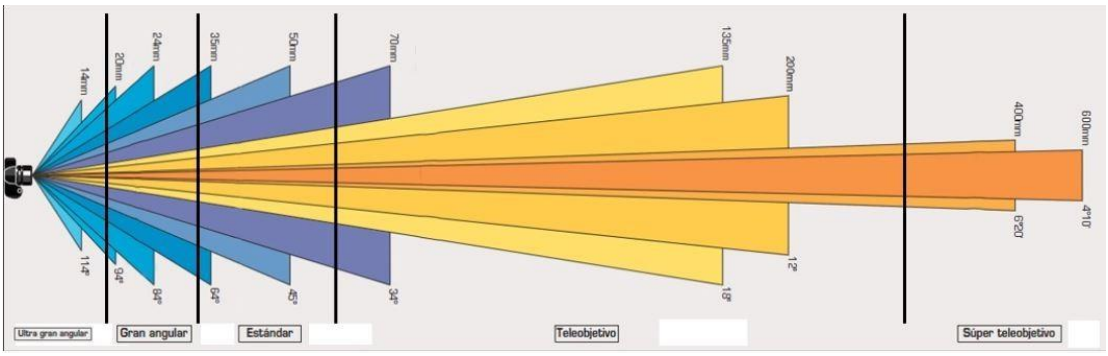


Figura 23: Ángulo de visión según la distancia focal y el tipo de lente [38]

Cuando hablamos de este tipo de lentes, u objetivos, se debe tomar en cuenta los términos: campo de visión, distancia focal y ángulo de visión. La distancia focal es la que existe entre el centro del lente y el punto donde se forma la imagen. El ángulo de visión indica la cantidad de escena que podemos abarcar en cada toma, depende del tamaño de la película y de la distancia focal del objetivo. A distancias focales más pequeñas se obtiene mayor ángulo de visión. El campo de visión es el área de visualización que una cámara puede capturar, puede calcularse con la distancia entre la cámara y el objeto y el ángulo de visión [38].

6.7.2. Especiftcaciones de la cámara digital

Cuadro 4: Especificaciones técnicas HD Webcam Logitech C270.

Tipo de conexión	USB 2.0
Tipo de enfoque	Fijo
Campo visual	60°
Longitud focal	4mm
Resolución óptica	1280x960
Captura de imágenes	320x240, 640x480, 800x600

6.7.3. Especiftcaciones del microscopio digital

Cuadro 5: Especificaciones técnicas Microscopio USB PCE-MM 800.

Tipo de conexión	USB 2.0
Rango de enfoque	10mm
Resolución	640x480
Fuente de iluminación	8 LED integrados
Zoom	200x, 500x, 800x, 1000x, 1600x
Dimensiones	110x33mm
Peso	90g

6.8. Escaneo láser 3D

Este tipo de tecnología, también conocida como lasergametría, consiste en el escaneo con un láser de los objetos obteniendo una nube de puntos a partir de la cual, con el software adecuado, se puede definir la forma y las dimensiones de los objetos. Se pueden llegar a encontrar millones de puntos, dependiendo de la resolución del escaneo láser. Permite una captura de información rápida, detallada y precisa de una superficie o volumen, siendo un método de medición no-intrusivo basado en la tecnología de escáner con láser pulsado. Este instrumento también se denomina habitualmente como Láser Escáner Terrestre o LIDAR terrestre [39].

6.8.1. Especiftcaciones del sensor GY-VL53L0X

Este sensor utiliza un láser de impulsos (pulsed time of flight), en el que se calcula la distancia a partir del tiempo de vuelo que tarda el retorno de la señal emitida por el láser.

Cuadro 6: Especificaciones técnicas sensor de distancia ToF GY-VL53L0X.

Láser	940nm marca VCSEL
Dimensiones	10.5x13.3mm
Rango de medición	30mm a 2m
Precisión	
Resolución	1mm
Interfaz de comunicación	I2C programable
Voltaje de alimentación	2.8V a 5V
Potencia de consumo	20mW

CAPÍTULO 7

Metodología

La investigación se realizó en la Reconstructora de Motores El Esfuerzo, kilómetro 167 carretera CA 2 Occidente, Cuyotenango, Suchitepéquez; en el área de rectificación de cilindros. Se realizó una investigación exhaustiva en internet sobre dispositivos similares en el mercado y proyectos que ayudaran a formular posibles alternativas para el diseño de prototipos. El procedimiento llevado a cabo se describe en el siguiente diagrama.

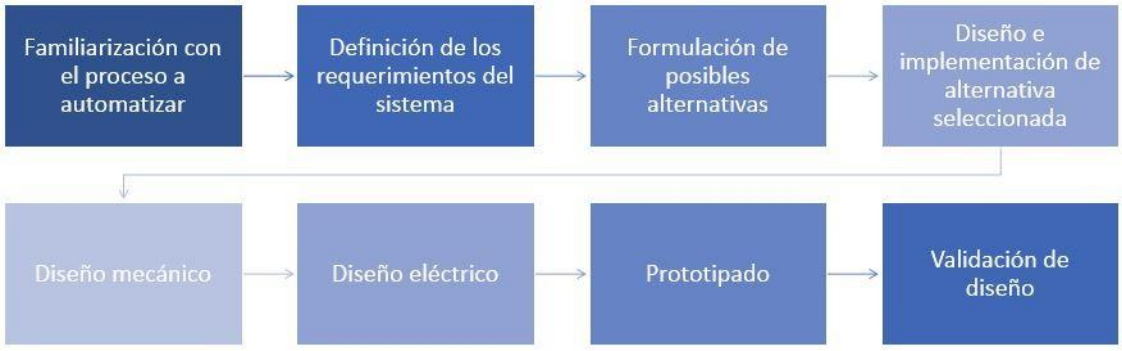


Figura 24: Procedimiento realizado para el diseño del dispositivo electrónico

7.1. Familarización con el proceso a automatizar

Como punto inicial del proyecto, se adquirió conocimiento del proceso que conlleva la inspección de las partes más importantes de un motor de un automóvil. Se prestó mayor interés en la inspección de los pistones, debido a que son una de las piezas involucradas en la conversión de energía de combustión a energía mecánica y porque son piezas relativamente pequeñas, con facilidad de manipulación.

Tras varias charlas con el personal encargado de inspeccionar dichas piezas, se procedió a buscar fuentes bibliográficas relacionadas con el tema, para un mejor conocimiento sobre el tecnicismo utilizado en la mecánica automotriz. Posteriormente se realizaron observaciones sobre el proceso de inspección, el cual se ha realizado por años de forma manual con la ayuda de un catálogo con todas las medidas estándar, no solo en dicha empresa sino que a nivel nacional. Dicho proceso se puede dividir en cuatro fases: identificación de la serie de motor, medición de diámetro, identificación de holgura excesiva en las ranuras de los anillos, identificación de averías.

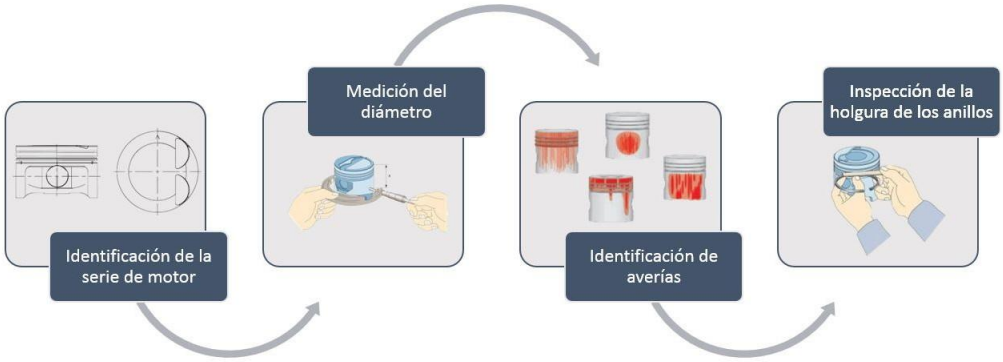


Figura 25: Fases del proceso de inspección de los pistones

7.2. Deftnición de los requerimientos del sistema

Para cada fase del proceso de inspección, se definieron los requerimientos necesarios para su posterior validación en un prototipo funcional. Además, debido a la gran variedad de marcas de automóviles en la actualidad, se ha delimitado la funcionalidad del sistema a los motores gasolina marca Toyota.

Para el despliegue de los resultados obtenidos de cada fase, se creó una interfaz gráfica. Además, en la misma interfaz se tienen botones de comandos para controlar remotamente el dispositivo electrónico.

Figura 26: Requerimientos de cada fase del proceso de inspección de un pistón

7.3. Formulación de posibles alternativas

Se pensó en un diseño que permitiera el posicionamiento adecuado del pistón para facilitar la toma de medidas e imágenes de cada fase del proceso de inspección. Para la medición de diámetros se consideró la idea de colocar dos sensores de imagen lineal en cada extremo de la circunferencia del pistón, tal y como se muestra en [40]. Una cortina láser se proyecta hacia el sensor y cuando un objeto impide el paso de una parte de esta cortina, la sombra se proyecta y puede calcularse su ancho dependiendo de la cantidad de fotodiodos que abarque [18].

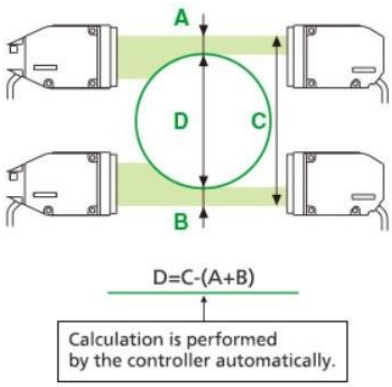


Figura 27: Aplicación micrómetro óptico con láser [40]

Por otra parte, se consideró en imitar el funcionamiento del dispositivo electrónico de medición y control de pistón que ofrece la empresa INFAS [2], o también el calibrador por divisiones de la empresa MGPL [3]. En ambos dispositivos se tienen dos palmadores, uno en cada extremo, que se acercan al pistón y, debido a su mecanismo de resorte interno que poseen, se contraen dando así una medida, que luego es procesada para hallar el diámetro exacto.

Figura 28: Dispositivo electrónico de medición y control de pistón de la empresa INFAS [2]

En la fase de identificación de holgura en las ranuras de los anillos se optó por implementar una cámara de escaneo de línea, cuyo princpio de funcionamiento se basa en el de los sensores de imagen lineal. Cada ranura será identificada por medio de pixeles, dependiendo de la cantidad de pixeles de la cámara así será la resolución de la medida. La empresa Keyence aplica este tipo de cámaras para la inspección de las ranuras, evitando las imperfecciones de manufactura en su geometría.

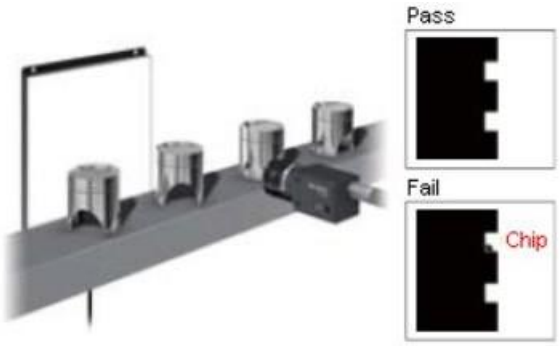


Figura 29: Inspección de defectos en las ranuras de los anillos de un pistón [41]

7.4. Diseño e implementación de alternativa seleccionada

De las dos alternativas para la medición del diámetro, se seleccionó la que incorpora un comparador digital, especificamente de la marca Mitutoyo [2]. Para la identificación de las ranuras, fue utilizado el módulo de cámara TSL1401CL [3], y para su medición se utilizó un módulo de sensor láser VL53L0X, ambos compatibles con Arduino.

Para la fase de reconocimiento de pistón, se requiere la adquisición de imágenes de la cabeza y la cara del pistón, por lo que se seleccionaron dos webcams Logitech C270. La ventaja de estas cámaras son su precio relativamente bajo, ofreciendo una resolución de imagen aceptable para esta aplicación. Por último, para la identificación de desgaste en la falda del pistón se utilizó un microscopio digital, debido al lente de gran aumento que posee y el sensor digital para la transferencia de las imágenes al ordenador.

Una vez seleccionada la alternativa que más se adecuaba a los requerimientos de cada fase del proceso de inspección, se procedió a desarrollar un diseño detallado de la estructura donde se montaron los instrumentos de medición y las cámaras, con su respectivo sistema electrónico.

7.5. Diseño mecánico

7.5.1. Mecanismo de desplazamiento lineal en un eje con husillo

Tomando en cuenta los requerimientos del sistema, se utilizó un mecanismo que permitiera el desplazamiento en un solo eje tanto del pistón como del comparador digital. Se adaptó el mecanismo que tienen las impresoras 3D para el posicionamiento del cabezal de extrusión. Para posicionar el pistón se utilizaron: dos barras lisas de 8mm, una barra roscada de 5mm, cuatro rodamientos LM8UU, un acoplador de eje de 3 mm a 5 mm, tornillos de 4mm y 3mm con sus respectivas tuercas y un motor paso a paso.

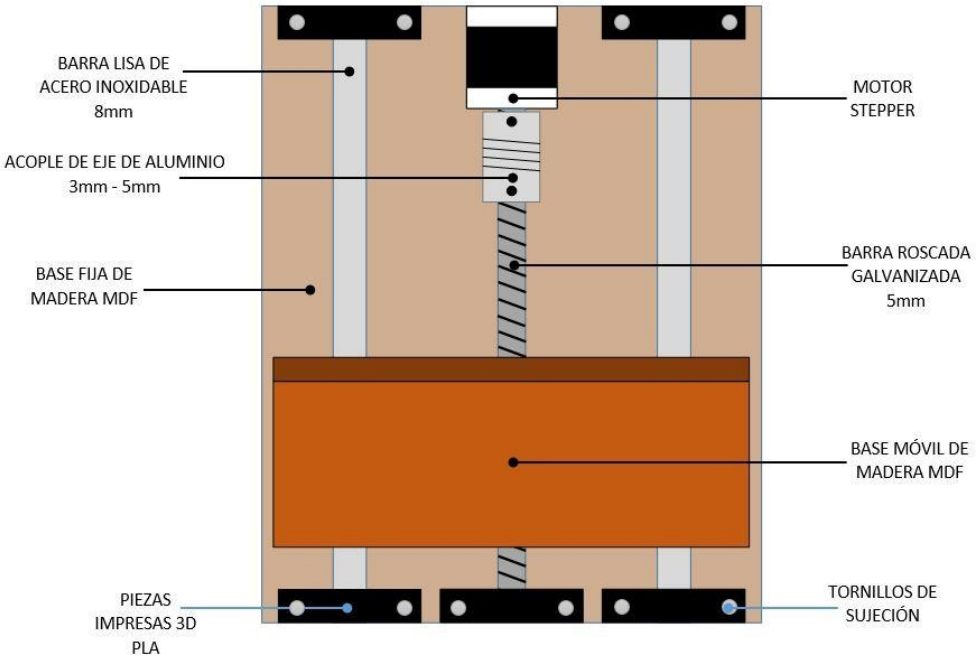


Figura 30: Diseño preliminar para el desplazamiento vertical del pistón

Para el mecanismo del desplazamiento horizontal del comparador digital y del microscopio digital se utilizaron: dos varillas de acero de 3mm, barra roscada de 5mm, un acoplador de eje de 3mm a 5mm, tornillos de 4mm y 3mm con sus respectivas tuercas y un motor paso a paso.

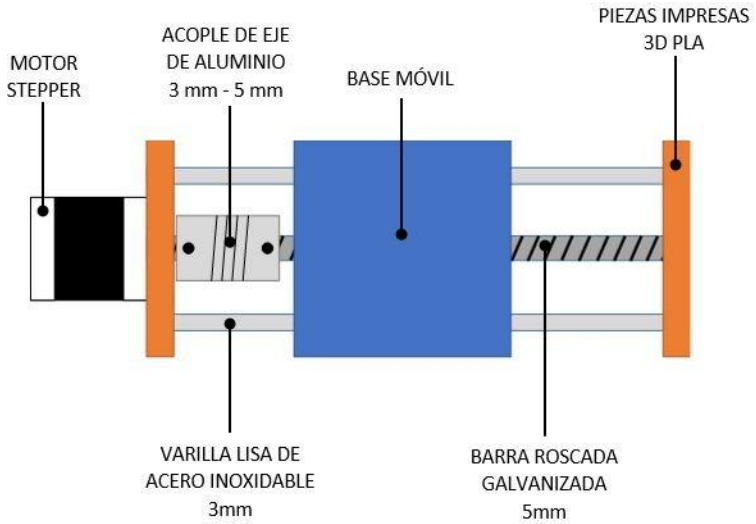
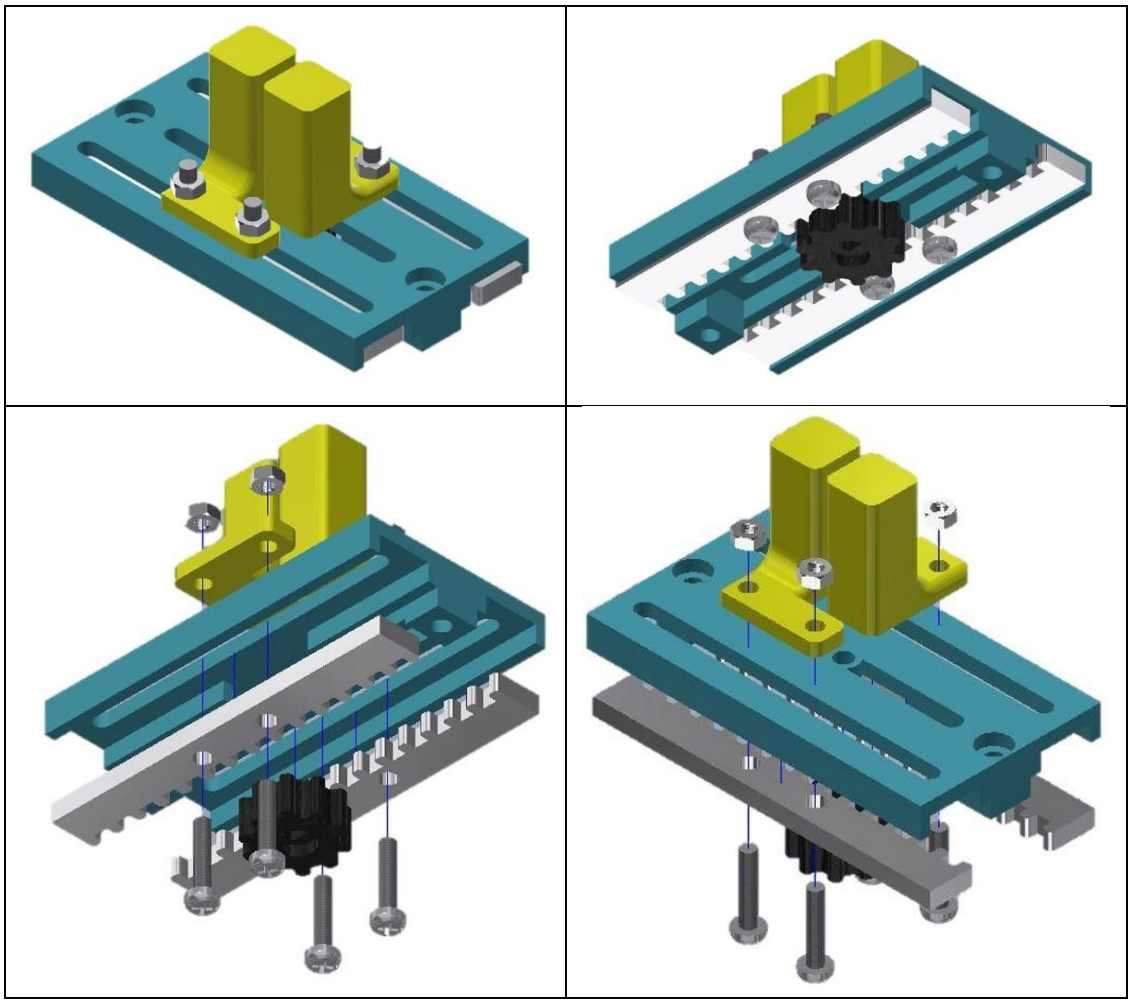


Figura 31: Diseño preliminar para el desplazamiento horizontal del comparador digital y del microscopio digital.

7.5.2. Selección del mecanismo para la sujeción del pistón

Lo que se buscaba con este mecanismo era sujetar el pistón en la posición como se muestra en el catálogo Teikin [42] o también como se encuentran dentro de los cilindros al estar armados [2]. Además la sujeción debía ser de tal manera que no impidiera efectuar las medidas y la toma de imágenes con las cámaras. Por lo que se propuso implementar un tipo mandril con dos mordazas (2-Jaw Chuck), como se muestra en la siguiente tabla.

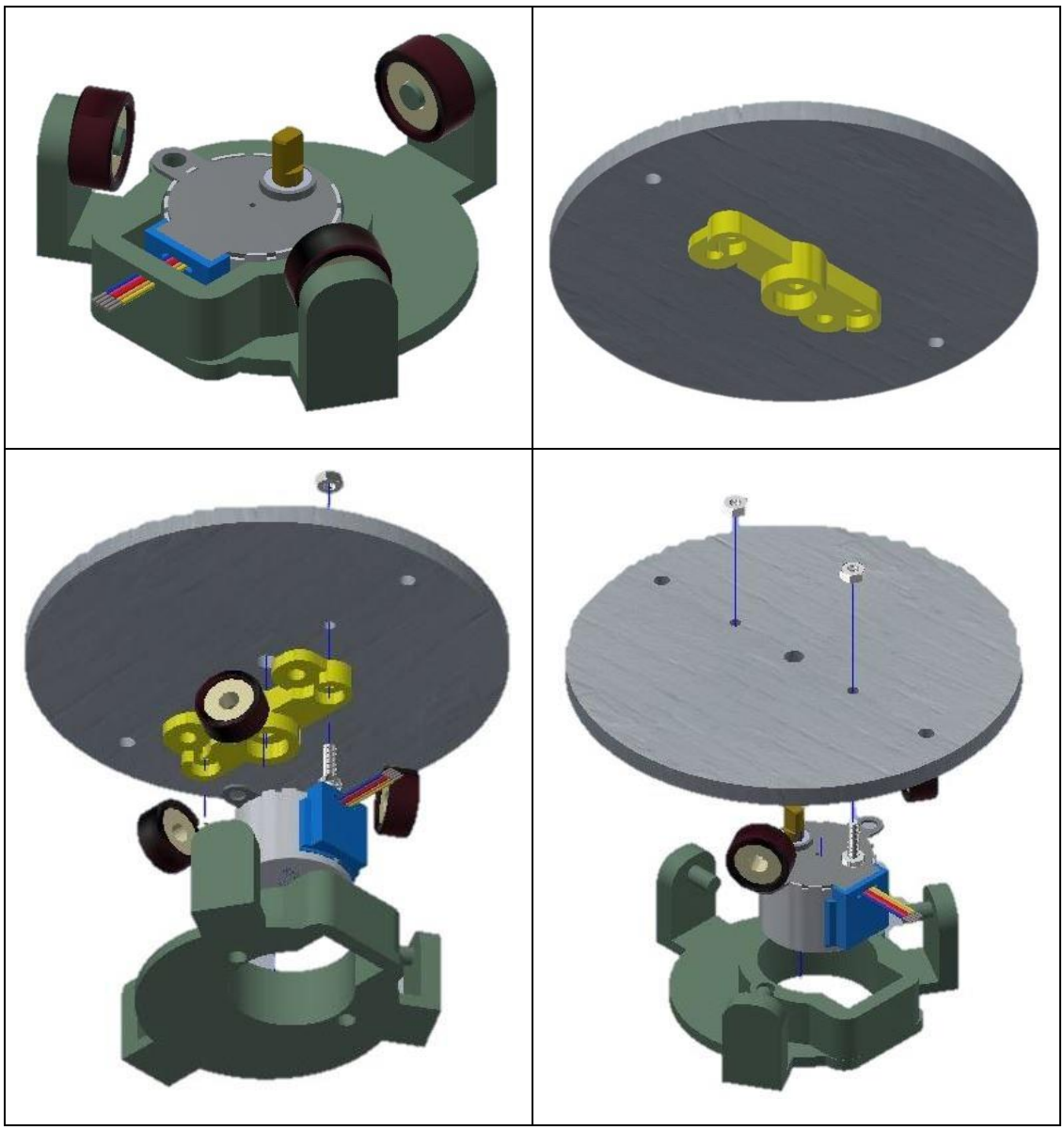
Cuadro 7: Diseño preliminar del mecanismo 2-Jaw Chuck. Vista superior, inferior y explosionada respectivamente



7.5.3. Mecanismo para el movimiento giratorio del pistón

Además de sujetar el pistón en la posición deseada, era necesario girarlo sobre su propio eje para permitir la toma de imágenes con el microscopio digital y las cámaras logitech. Se propuso entonces imitar el diseño de las bases giratorias utilizadas para escaneo 3D. Para este diseño fue necesario utilizar rodamientos de 5mm x 16mm.

Cuadro 8: Diseño preliminar de la base giratoria. Vista superior, vista inverior y vista explosionada.



7.5.4. Acople de lentes de enfoque para webcams

Debido a que las webcams poseen un enfoque fijo [4], no era posible enfocar el pistón a una corta distancia. Además, el campo de visión es muy pequeño, por lo que no se lograba capturar una buena imagen de la pieza completa. Se recurrió al uso de lentes ópticos para solucionar el problema. Fueron necesarios dos lentes gran angular y dos lentes macro. Los primeros ayudaron a aumentar el campo de visión y los otros a obtener un mejor enfoque. Para posicionarlos de forma fija frente al lente de cada cámara, se diseñaron piezas que se acoplaran a ambas geometrías.

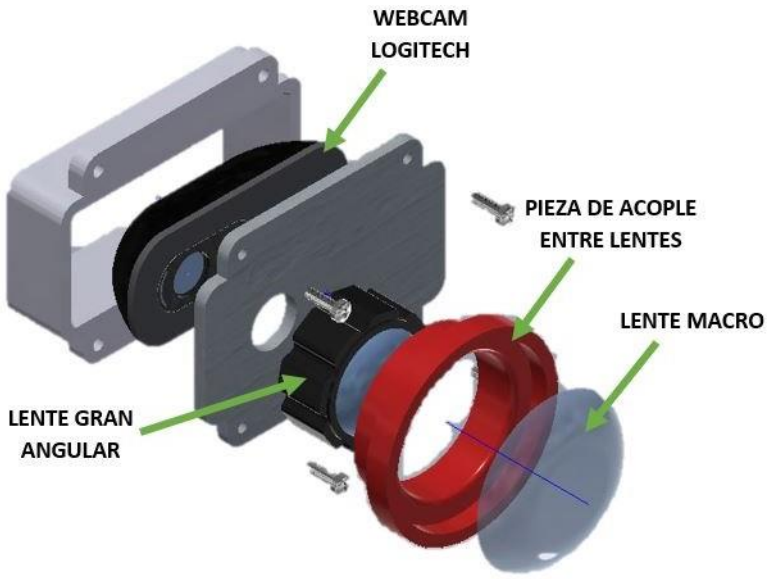


Figura 32: Acoplamiento del lente gran angular y macro a webcam

7.5.5. Diseño de piezas para impresión en 3D y corte láser

Se utilizó el software Autodesk Inventor 2018 para el diseño de las piezas personalizadas que complementan los mecanismos descritos anterioremente. Luego, cada pieza se exportó al software Repetier Host como archivo .stl para su fabricación en una impresora 3D, utilizando material PLA. Además, también se fabricaron piezas en 2D cortadas en láser en tablas de material mdf de 3mm y 5mm de grosor.

7.6. Diseño eléctrico

7.6.1. Selección de instrumentación

A continuación se detalla la lista de electrónicos utilizados y su funcionamiento dentro del dispositivo:

- Arduino Mega 2560 Rev3: Interfaz de comunicación y programación de los sensores y actuadores del dispositivo.
- RepRap Pololu Arduino Mega Shield (RAMPS): Control de la etapa de potencia y de los drivers de los motores paso a paso.
- RepRap Discount Smart Controller: Interfaz de control remoto entre el usuario y el dispositivo, permitiendo la selección de la acción a realizar. Incorpora una pantalla LCD de 20x4, lectora de SD y un encoder rotativo.
- Módulo A4988: Control de los motores paso a paso en conjunto con la RAMPS.

- Sensor VL53L0X: Medición de las ranuras de los anillos del pistón por medio de un barrido vertical láser.
- Cámara de escaneo de línea TSL1401CL: Detección y medición aproximada de las ranuras de los anillos del pistón.
- Stepper Nema 17: Movimiento lineal de cualquier tipo de pistón sin presentar problemas por su peso.
- Interruptor final de carrera: Delimitan la posición de la base móvil del comparador digital, la base móvil del microscopio digital, el mecanismo 2-Jaw Chuck, la base giratoria del pistón y la base móvil del pistón.
- Motor 28BYJ-48: Movimiento giratorio de la base del pistón y movimiento lineal del mecanismo 2-Jaw Chuck.
- Stepper Mitsumi M35SP-8N: Movimiento lineal de la base móvil del comparador digital y el microscopio digital.

7.6.2. Módulo de comunicación Mitutoyo-Arduino

Los instrumentos de medición digimáticos de la marca Mitutoyo tienen sus propios cables para la transferencia de los datos de medición a dispositivos externos, ya sean pantallas, procesadores o microcontroladores. Se utilizó un cable de conexión SPC con salida IDC de 10 pines, para adaptarlo al Arduino se realizó un pequeño circuito. El microcontrolador le proporciona una señal de reloj que indica los pulsos para la transferencia de datos.

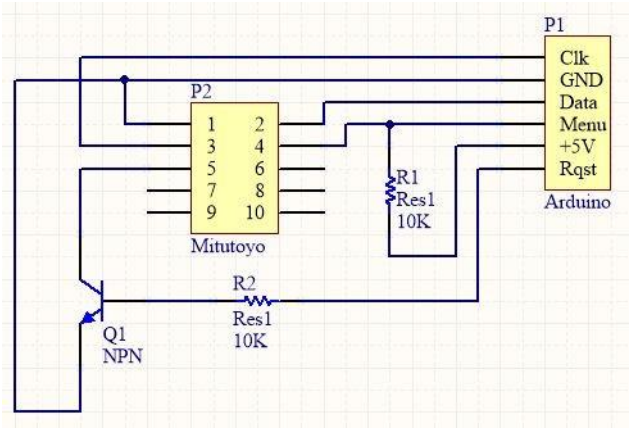


Figura 33: Esquemático interfaz de comunicación del comparador Mitutoyo con Arduino.

7.6.3. Calibración de los drivers A4988

El driver A4988 incorpora unos pines (MS1, MS2, MS3) que se utilizan para configurar la resolución de los pasos en los que se mueve el motor. Para esta aplicación, se utilizaron las resoluciones que se muestran en la tabla 10 . Para hacer girar 360° los motores seleccionados,

se debe entregar una determinada cantidad de pulsos, dependiendo del ángulo de paso de cada motor.

$$V_{ref} = 8 \cdot I_{max} \cdot R_{cs} \quad (4)$$

Cuadro 9: Resolución de micro pasos para el driver A4988.

MS1	MS2	MS3	Resolución
Low	Low	Low	Full step
High	Low	Low	Half step
Low	High	Low	Quarter step
High	High	Low	Eighth step
High	High	High	Sixteenth step

Cuadro 10: Resolución de pasos por cada motor.

Motor	Resolución	Ángulo de paso	Pulsos por vuelta
Nema 17	Sixteenth step	1.8°	3200
Mitsumi M35SP-9	Sixteenth step	7.5°	768
28BYJ-48	Quarter step	0.088°	16364

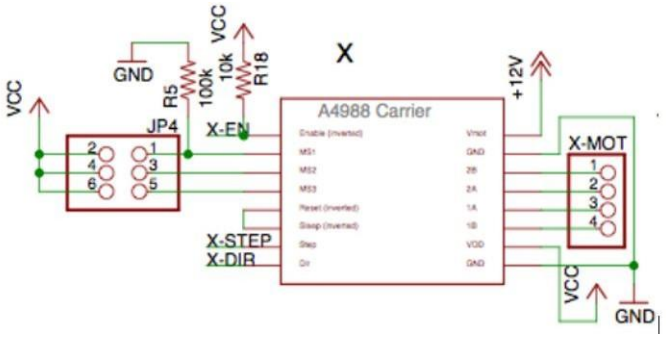


Figura 34: Esquematico del driver A4988 en la RAMPS v1.4 (Tomado de la página oficial de RepRap).

Para que el motor no pierda pasos ni se pueda quemar por algún exceso de corriente, es necesario ajustar el voltaje de referencia del driver. Este valor se calcula con la ecuación 4 que se presenta en la hoja de datos del driver, conociendo la corriente máxima de cada motor y el valor de las resistencias de sensibilidad del mismo driver.

Cuadro 11: Voltaje de referencia de cada motor para un Rcs de 0.1.

Motor	Corriente máxima	Corriente de operación	Voltaje de referencia
Nema 17	1.7A	1A	800mV
Mitsumi M35SP-9	259mA	200mA	160mV
28BYJ-48	500mA	200mA	160mV

7.6.4. Sistema de iluminación

Se utilizaron aros de luz de 35 LEDs para selfie y fueron adaptados para iluminar el campo de visión de las cámaras Logitech. Los aros se conectaron a 5V con una resistencia de 100 ohmios, obteniendo una corriente de 50mA. Además, se utilizaron dos diodos LED para la detección de líneas de la cámara TSL1401CL.

7.6.5. Diagrama eléctrico

Se presenta un diagrama de las conexiones de los distintos sensores y actuadores con la interfaz RAMPS v1.4. Cabe mencionar que el Arduino Mega se encuentra ya incorporado en el diagrama junto con la RAMPS, el cual permite la comunicación por USB con el ordenador. Las cámaras Logitech junto con el microscopio se conectan también por medio de USB directamente al ordenador.

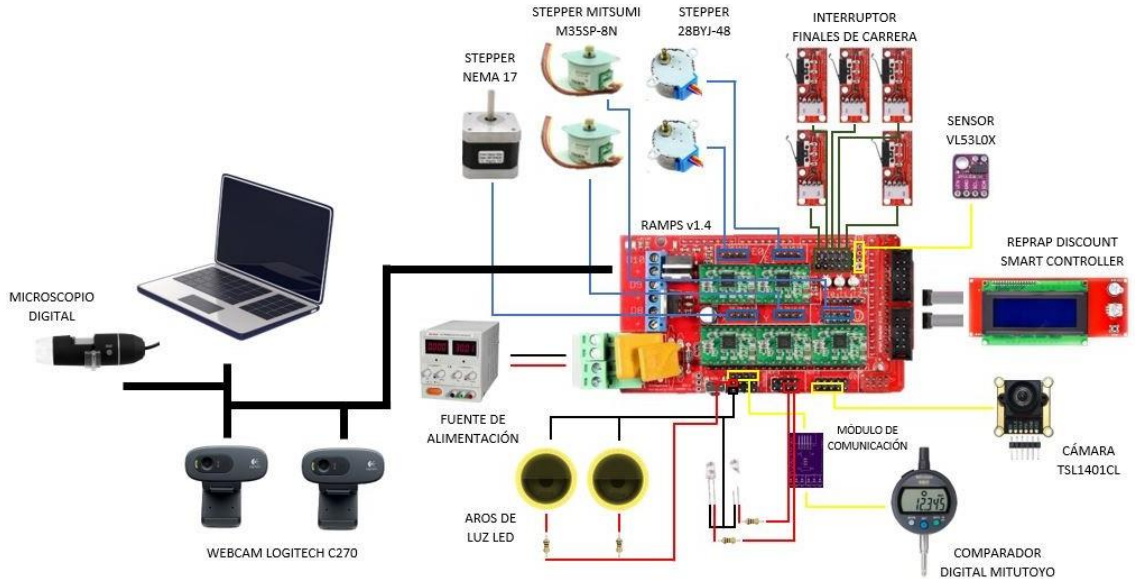


Figura 35: Diagrama eléctrico del prototipo final

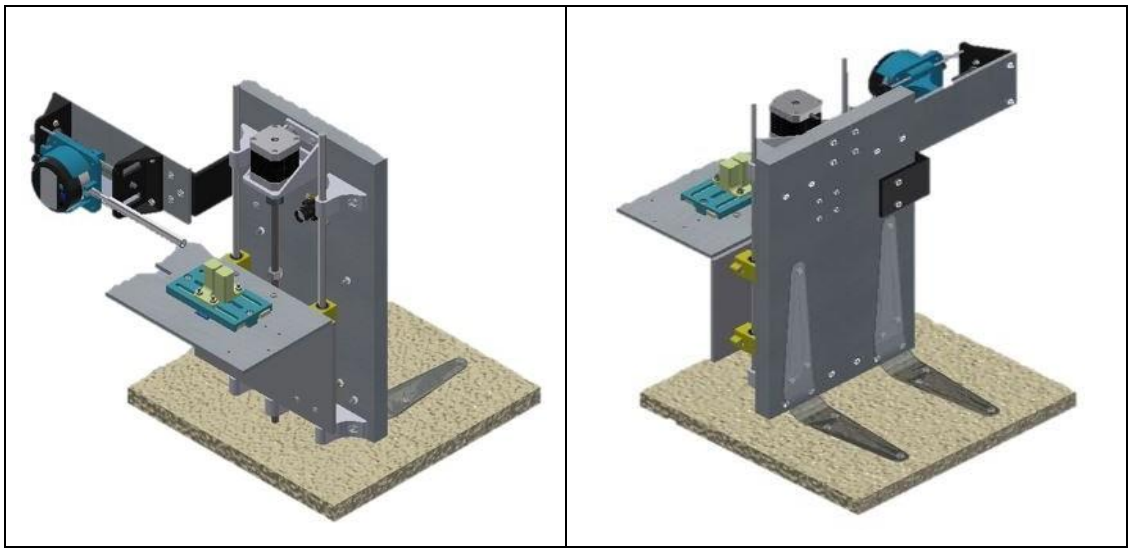
7.7. Prototipado del dispositivo

7.7.1. Prototipo 1

Para este primer diseño se buscaba tomar mediciones de los diámetros con el comparador digital, posicionar de forma correcta el pistón en el mecanismo 2-Jaw Chuck, desplazar verticalmente el piston y obtener una medición aproximada de las ranuras de los anillos con la cámara TSL1401CL. Todos los mecanismos descritos anteriormente se montaron sobre una base sólida de madera con tornillos y un par de vigas de metal para estantes de madera.

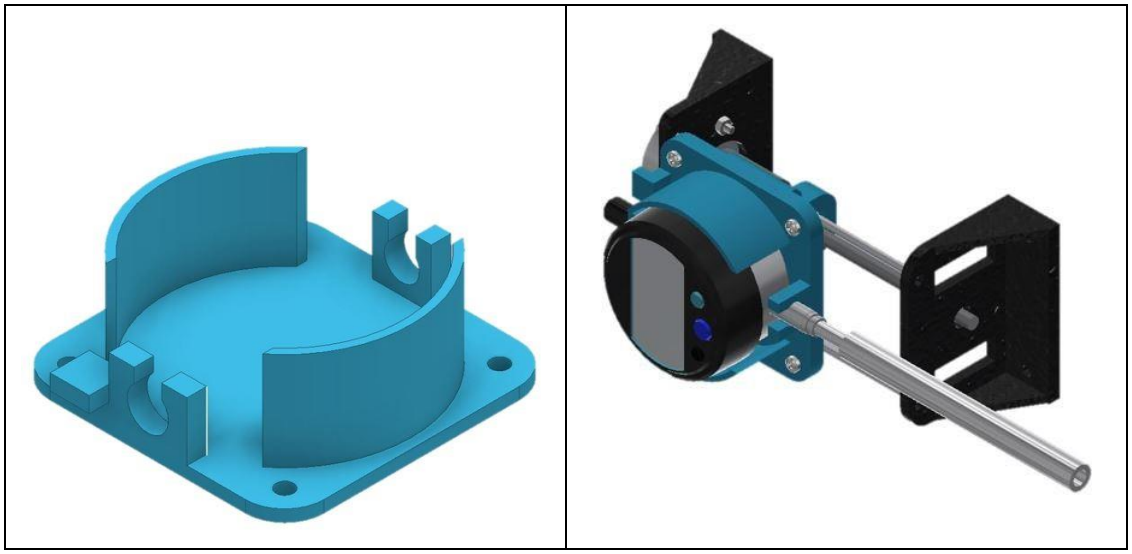
Es importante mencionar que se imprimió una extensión en PLA para el palpador del comparador Mitutoyo con el objetivo de tocar el pistón y registrar las medidas.

Cuadro 12: Diseño preliminar del prototipo 1. Vista frontal y trasera.



Para montar el comparador en el mecanismo de desplazamiento lineal, se diseñó una pieza que se adaptara a su geometría. A continuación se presenta el diseño de dicha pieza junto con el ensamble del mecanismo.

Cuadro 13: Mecanismo desplazamiento lineal del comparador Mitutoyo del prototipo 1.



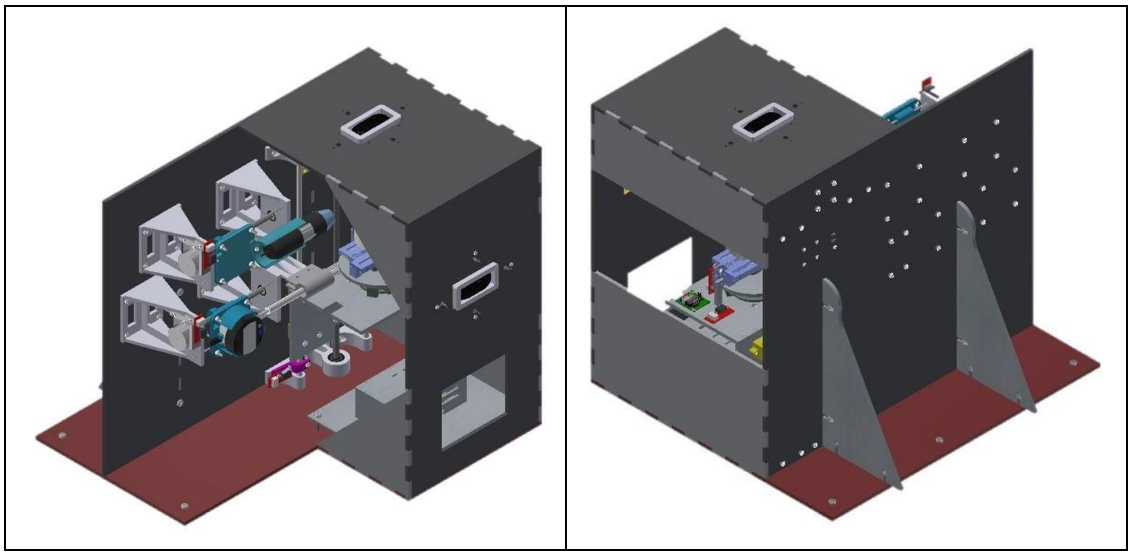
7.7.2. Prototipo 2

A diferencia del primer diseño, se lograron montar las cámaras de tal manera que capturan imágenes de la cabeza y la parte frontal del pistón. Para la base del mecanismo de

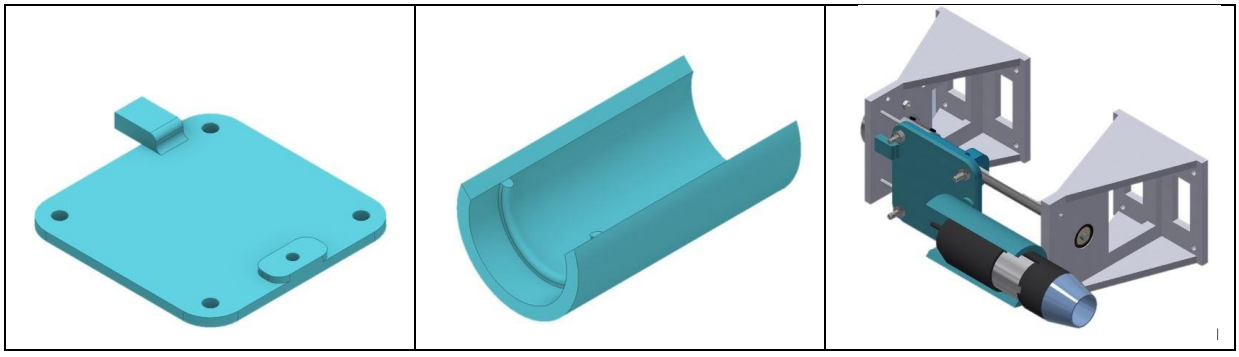
sujeción, se le incorporó movimiento circular añadiendo el mecanismo de giro propuesto anteriormente. El microscopio digital está montado sobre una base similar al del comparador, con una pequeña modificación que se adapa a su geometría circular. Este prototipo ofrece una mayor estabilidad de todos los componentes y mejor control de su posicionamiento dentro del espacio de trabajo.

La extensión de PLA se reemplazó por otra de acero inoxidable, sometida un proceso de torneado. Las vigas de metal fueron reemplazadas por vigas de madera personalizadas y cortadas en láser. Debido al peso de la varilla de acero, debía diseñarse una pieza auxiliar que ayudara a soportar el peso y, al mismo tiempo, guiar el movimiento lineal del comparador digital.

Cuadro 14: Diseño preliminar del prototipo 2. Vista frontal y trasera.

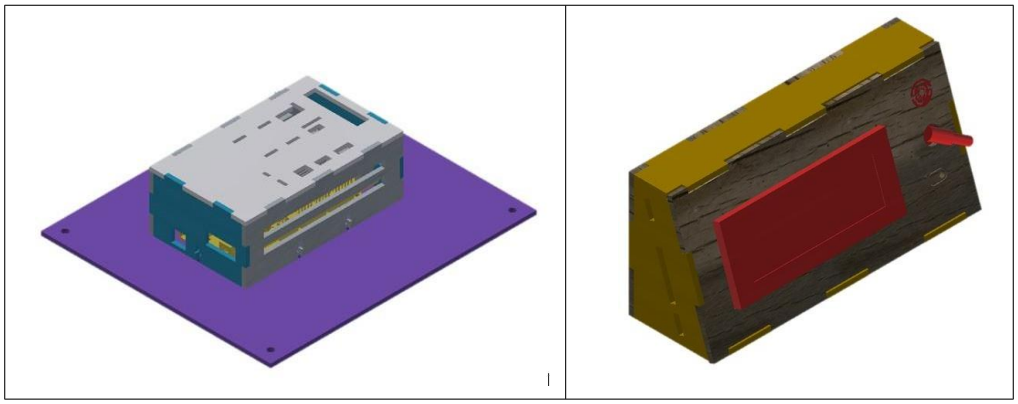


Cuadro 15: Mecanismo desplazamiento lineal del microscopio USB.



Se añadieron rodamientos de bolas en las piezas que sujetan las varillas roscadas para que su movimiento sea lo más suave posible; el microcontrolador se colocó junto con la RAMPS en un compartimiento aislado; la pantalla LCD se colocó en una recámara, diseñada por el operario del makerlab de la UVG.

Cuadro 16: Diseño de recámaras para el microcontrolador Arduino y la pantalla LCD.



CAPÍTULO 8

Técnicas de medición y posicionamiento de los sensores e instrumentos de medición

La cantidad de pasos por mm para los mecanismos de desplazamiento del pistón, comparador y microscopio se calcularon con la ecuación 5 y conociendo la configuración de micropasos en los drivers A4988 de la Tabla 9. Para el comparador y el microscopio se están utilizando varillas métricas M6, mientras que para el pistón una varilla métrica M8.

$$P_{pm} = \frac{P_{pv} \cdot M_{driver}}{P_{varilla}} \tag{5}$$

Cuadro 17: Cantidad de pasos por milímetro de los motores del pistón, comparador y microscopio.

Motor	**Pasos/vuelta**	**Paso varilla**	**Micropasos driver**	**Pasos/mm**
Nema 17	3200	1.75	16	1828
Mitsumi M35SP-9	768	1	16	768

8.1. Medición del diámetro con el comparador Mitutoyo

Conociendo cómo se mide el diámetro de un pistón con un micrómetro convencional, se adaptó esta técnica con un comparador digital. El pistón se posicionó de tal forma que la falda del pistón quedara perpendicular al eje de desplazamiento del palpador. Al acercar el palpador al pistón, se registra una medida d en milésimas de milímetro, con un rango de medición de hasta 13.12mm. El diámetro del pistón es calculado entonces en milímetros con la ecuación 6.

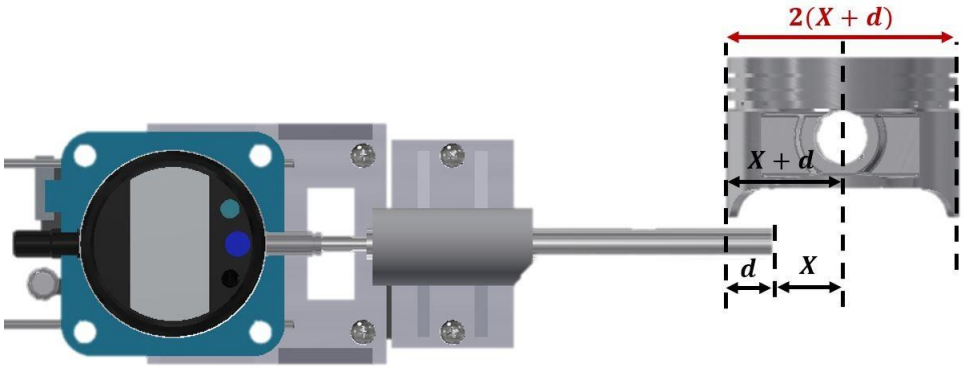


Figura 36: Cálculo de la posición relativa del comparador digital Mitutoyo respecto al pistón.

$$D_{piston} = 2 \cdot (d_{Mit} + X) \tag{6}$$

La distancia X es una medida constante que fue establecida tomando en consideración el rango de medición del comparador y el rango de los diámetros de los pistones Toyota (69-100mm), se encontró entonces un valor mínimo de 34mm. Sin embargo, con estos valores no es posible medir diámetros mayores a 94mm por el pequeño rango de medición que ofrede el comparador.

8.2. Posicionamiento del microscopio respecto al pistón

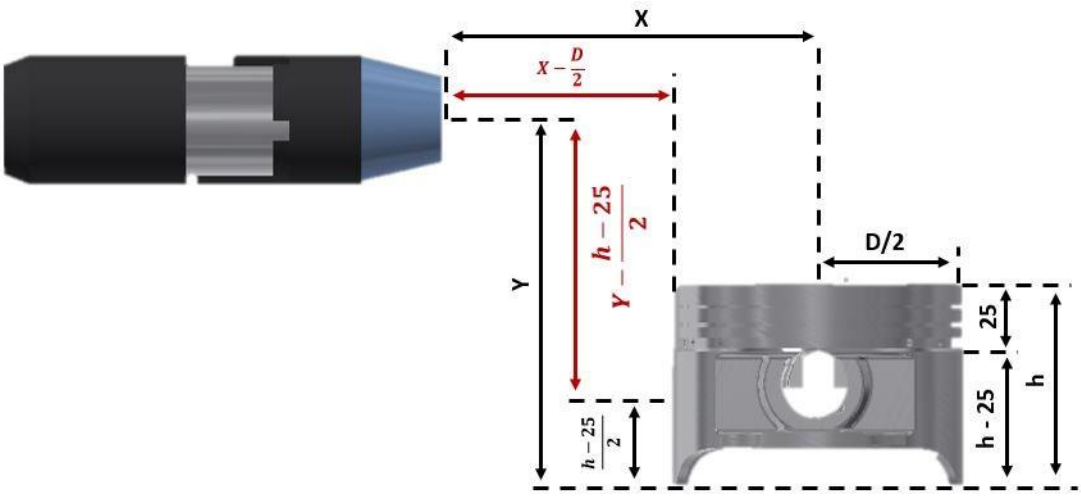


Figura 37: Cálculo de la posición relativa del microscopio respecto al pistón.

El objetivo de este cálculo era encontrar una determinada posición que permitiera la captura de imágenes de la falda del pistón y que se acoplara a los distintos diámetros de los pistones. Era necesario un cálculo de posición vertical de la base móvil del pistón y uno de posición horizontal de la base móvil del microscopio.

$$x_{mic} = X - \frac{D_{piston}}{2} - R_{enfoque} \tag{7}$$

Debido a que la fase de idenificación de desgaste tiene lugar después del reconocimiento de la serie del pistón y la medición con el comparador, se conoce la altura total y el diámetro del pistón. En la ecuación 7 se considera el rango de enfoque mínimo del microscopio, según la tabla 5. La distancia X es una medida constante de la posición relativa del lente del microscopio respecto al centro del pistón dentro del prototipo final.

$$y_{piston} = Y - \frac{h_{piston} - 25}{2} \tag{8}$$

Para cálculo de la posición vertical de la base móvil del pistón se estableció una constante de 25mm, que representa un ancho promedio de lo que abarcan las ranuras del pistón en su geometría. La distancia Y es una medida constante de la posición relativa de la base móvil del pistón respecto al centro del microscopio dentro del prototipo final.

8.3. Posicionamiento del pistón respecto a la cámara de escaneo de línea

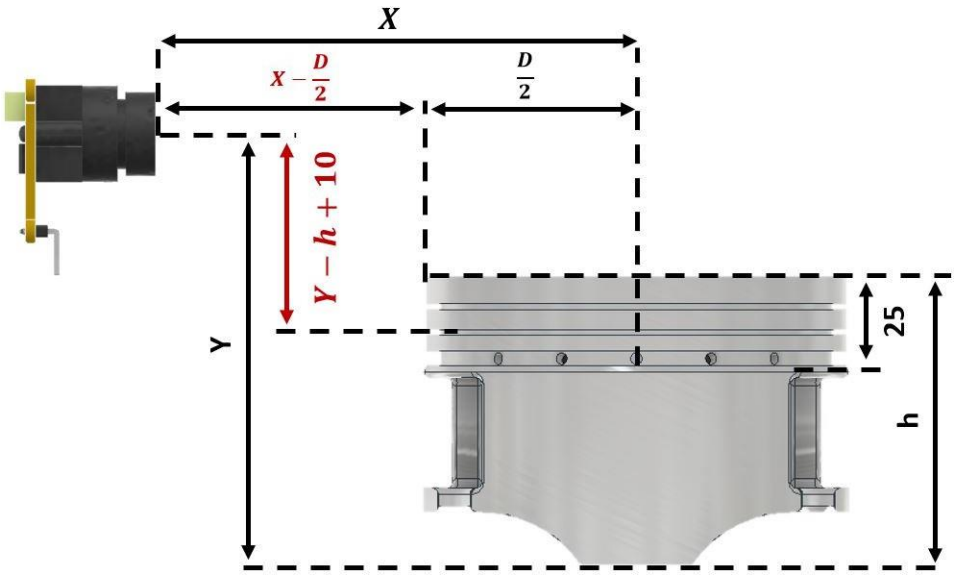


Figura 38: Cálculo de la posición relativa del pistón respecto a la cámara TSL1401CL dentro del área de trabajo.

Para posicionar el área de las ranuras del pistón dentro del campo de visión de la cámara, era necesario encontrar una ecuación que relacionara la altura total del pistón y la posición vertical de la cámara respecto a la base móvil del pistón. La distancia Y es una medida constante que se estableció luego de ensamblar el prototipo final.

$$y_{camara} = Y - h_{piston} + 12.5 \tag{9}$$

Además, se calculó la distancia horizontal del lente de la cámara hacia la superficie del pistón, cuyo valor varía dependiendo del diámetro de cada pistón. Más adelante se usó este valor para encontrar el tamaño del campo de visión de la cámara, en función de su distancia focal y el ángulo de visión.

$$x_{lente} = X - \frac{D}{2} \tag{10}$$

8.4. Posicionamiento del pistón respecto al sensor láser

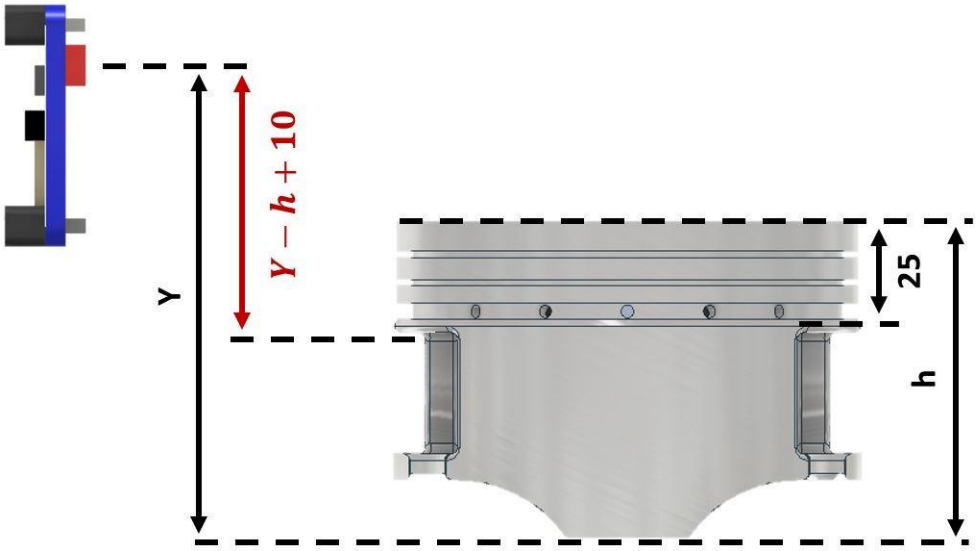


Figura 39: Cálculo de la posición relativa del pistón respecto al sensor láser VL53L0X dentro del área de trabajo.

Para posicionar el pistón frente al emisor láser del sensor de distancia, se encontraron las distancias relativas del pistón respecto al sensor, de la misma forma como se hizo con la cámara mencionado anteriormente. Con la diferencia que ahora se comienza desde abajo de la primer ranura del tercer anillo.

$$y_{laser} = Y - h_{piston} + 10 \tag{11}$$

CAPÍTULO 9

Métodos de visión por computador para el reconocimiento de pistones

9.1. Mediante Image Hashing y detección de bordes con Python y OpenCV

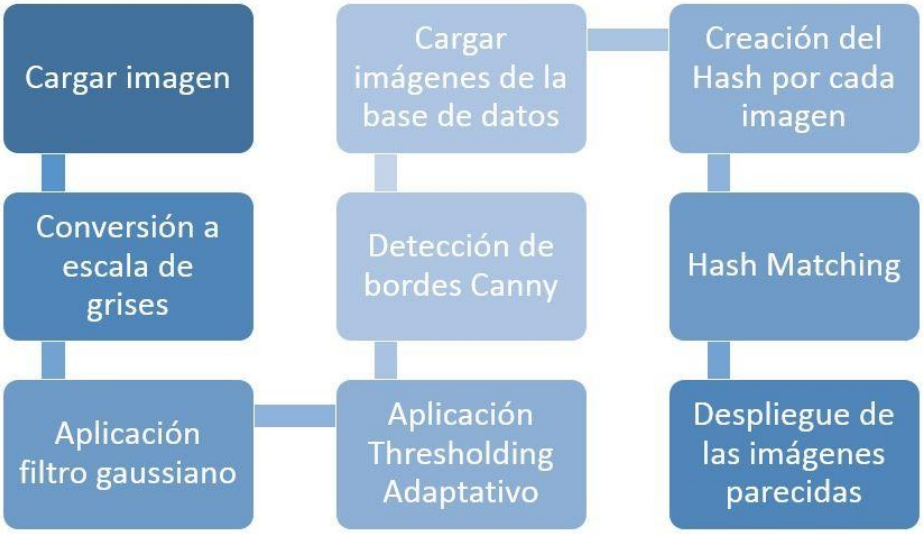


Figura 40: Etapas del reconocimiento de pistones mediante Image Hashing y detección de bordes.

Con este método se buscaba comparar los pistones de muestra con las imágenes que se presentan en el catálogo Teikin. Debido a que el catálogo solo proporciona una imagen de los bordes de la forma geométrica del pistón, era necesario encontrar los bordes de los pistones de muestra. Esto se realizó con los métodos de visión artificial de la librería OpenCV en

el entorno de programación de Python. Después de haber encontrado los bordes se debía comparar la imagen resultante con las imágenes del catálogo, por lo que se extrajeron de forma manual las imágenes de las 140 series de motor. Luego, para cada imagen se calculó su valor hash mediante Image Hashing. Con este valor numérico ya era posible comparar las imágenes para determinar que tanto se parecían entre sí.

9.1.1. Codiftcación del algoritmo de reconocimiento mediante Image Hashing y detección de bordes

```
#Importacion de librerias para el procesamiento de imagenes
import cv2
import imagehash
import numpy
from PIL import Image

#Cargando la imagen de muestra
imagenM = cv2.imread('D:/Documentos/Mis Documentos/Trabajo de Graduacion/'\
                     'Base de Datos/Muestra/Cara/1ZZFE.jpg')

#Conversion a escala de grises
imagenMGray = cv2.cvtColor(imagenM, cv2.COLOR_BGR2GRAY)

 #Aplicacion de filtros gaussianos
imagenMBlur = cv2.GaussianBlur(imagenMGray, (5,5), 0)
imagenMBlur = cv2.GaussianBlur(imagenMBlur, (5,5), 0)
imagenMBlur = cv2.GaussianBlur(imagenMBlur, (5,5), 0)
imagenMBlur = cv2.GaussianBlur(imagenMBlur, (5,5), 0)
imagenMBlur = cv2.GaussianBlur(imagenMBlur, (5,5), 0)
imagenMBlur = cv2.GaussianBlur(imagenMBlur, (5,5), 0)
imagenMBlur = cv2.GaussianBlur(imagenMBlur, (5,5), 0)
imagenMBlur = cv2.GaussianBlur(imagenMBlur, (5,5), 0)

#Aplicacion de thresholding adaptativo
imagenMThresh = cv2.adaptiveThreshold(imagenMBlur,255,cv2.ADAPTIVE_THRESH_MEAN_C,
                                      cv2.THRESH_BINARY,11,2)

#Deteccion de bordes con el metodo Canny
imagenMCanny = cv2.Canny(imagenMThresh, 10, 15)

#Cargando las imagenes de la base de datos
i = 1
DBC = []
while i <= 140:
    DBC.append("D:\\Documentos\\Mis Documentos\\Trabajo de Graduacion"\
               "\\Base de Datos\\Cara\\"+str(i)+".jpg")
    i += 1
```

```
#Creacion del valor hash de cada imagen en la base de datos
imagesC = {}
hashesC = []
for image in DBC:
    try:
        hash = imagehash.phash(Image.open(image))
        hashesC.append(hash)
    except Exception as e:
        print("Problema ",e," con ",image)
    imagesC[hash] = imagesC.get(hash, []) + [image]

#Conversion image cv2 a image pil
imagenMConvert =  Image.fromarray(imagenMThresh)

#Creacion del valor hash de la imagen de muestra
imagenMHash = imagehash.phash(imagenMConvert)

#Se encuentran las imagenes parecidas a la imagen de muestra (Hash matching)
i = 0
hashMatching =[]
while  i<140:
    if (abs(imagenMHash-hashesC[i])<30):
        hashMatching.append(str(i+1)+".jpg")
    i+=1

#Despliegue de las imagenes parecidas
for numero in hashMatching:
    imagenH = cv2.imread('D:/Documentos/Mis Documentos/Trabajo de Graduacion/'\
                         'Base de Datos/Cara/'+numero)
    cv2.imshow("Imagen "+numero, imagenH)
```

9.2. Mediante Feature Points en MATLAB

Figura 41: Etapas del reconocimiento de pistones mediante Feature Points.

Este método extrae los rasgos de la superficie de los objetos de interés de una imagen de muestra y luego son comparados con los rasgos de la imagen en la base de datos. Debido a que los métodos de procesamiento de imagen de Matlab solo admiten imágenes en escala de grises, se le aplicó una conversión a las dos imágenes a comparar. La figura 41 resume el pseudocódigo de la técnica aplicada y a continuación se encuentra el código empleado en el entorno de programación de Matlab.

9.2.1. Codiftcación del algoritmo de reconocimiento de pistones mediante Feature Points

```
%Cargando la imagen de la base de datos
imagenBD = imread('D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\dataBase\3VZE\recorteB16.jpg');

%Cargando la imagen de muestra
imagenM = imread('D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\dataBase\3VZE\recorteB20.jpg');

%Conversión a escala de grises
imagenBDGray = rgb2gray(imagenBD);
imagenMGray = rgb2gray(imagenM);

%Detección de los Surf Features
imagenBDPts = detectSURFFeatures(imagenBDGray);
```

```
imagenMPts =  detectSURFFeatures(imagenMGray);

%Extracción de los Surf Features
[imagenBDFeatures, imagenBDValidPts] = extractFeatures(imagenBDGray, imagenBDPts);
[imagenMFeatures, imagenMValidPts] = extractFeatures(imagenMGray, imagenMPts);

%Matching  Features
index_pairs = matchFeatures(imagenBDFeatures, imagenMFeatures);

imagenBDMatchedPts = imagenBDPts(index_pairs(:,1)).Location;
imagenMMatchedPts = imagenMPts(index_pairs(:,2)).Location;

%Seleccionando los Features dentro de  los  objetos
gte = vision.GeometricTransformEstimator;
gte.Method = 'Random Sample Consensus (RANSAC)';
[tform_matrix, inlierIdx] = step(gte, imagenBDMatchedPts, imagenMMatchedPts);

imagenBDInlierPts = imagenBDMatchedPts(inlierIdx, :);
imagenMInlierPts = imagenMMatchedPts(inlierIdx, :);

%Mostrando resultados
figure;
showMatchedFeatures(imagenM, imagenBD, imagenMInlierPts, imagenBDInlierPts,...
'montage');
title('Showing match only with Inliers');
```

9.3. Mediante Machine Learning con MATLAB

Figura 42: Etapas del reconocimiento de pistones mediante Machine Learning.

Para este método es necesario utilizar el Computer Vision Toolbox de Matlab. Lo que se busca es entrenar clasificadores que permitan reconocer diferentes tipos de pistones según la serie, extrayendo los rasgos con histogramas de gradientes (HOG Features) de las imágenes para utilizarlos como datos de entrenamiento. A diferencia de los métodos anteriores, se debe contar con suficientes imágenes de entrenamiento de diferentes muestras de una misma serie para que la exactitud del clasificador sea lo más alto posible. Los modelos clasificadores se crearon con la ayuda del Classification Learner App de Matlab, obteniendo así la codificación de cada uno y permitiendo su uso en cualquier parte del programa principal para crear nuevos clasificadores.

Tanto para la cabeza como la cara del pistón, la codificación es similar. La diferencia está en la cantidad de rasgos extraídos debido a los distintos tamaños de imagen. Para la cabeza se tiene un tamaño de imagen de 676x671 con un total de 13680 rasgos, mientras que para la cara se tiene un tamaño de 626x801 con 15552 rasgos.

9.3.1. Codiftcación del modelo clasiftcador de la cabeza del pistón

```
function [trainedClassifier, validationAccuracy] = trainClassifierBase(trainingData)

inputTable = trainingData;
numIms = 24; %CANTIDAD DE IMAGENES POR CADA PISTON EN LA BASE
largoTabla = length(inputTable.Properties.VariableNames);
predictorNames = inputTable.Properties.VariableNames(1:largoTabla-1);
predictors = inputTable(:, predictorNames);
response = inputTable.objectType;
isCategoricalPredictor = [];
for k=1:largoTabla-1
    isCategoricalPredictor(k) = 0;
end

seriesLabels = [];
seriesLabelsC = categorical(seriesLabels);
for k=1:largoTabla/numIms
    seriesLabelsC(k,1) = inputTable.objectType(k*numIms);
end

template = templateSVM(...
    'KernelFunction', 'linear', ...
    'PolynomialOrder', [], ...
    'KernelScale', 'auto', ...
    'BoxConstraint', 1, ...
    'Standardize', true);
classificationSVM = fitcecoc(...
    predictors, ...
    response, ...
    'Learners', template, ...
    'Coding', 'onevsone', ...
```

```
    'ClassNames', seriesLabelsC);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
svmPredictFcn = @(x) predict(classificationSVM, x);
trainedClassifier.predictFcn = @(x) svmPredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct
trainedClassifier.RequiredVariables = {'trainingFeaturesBase1', ...
'trainingFeaturesBase13680'};
trainedClassifier.ClassificationSVM = classificationSVM;
trainedClassifier.About = 'This struct is a trained model exported from ...
Classification Learner R2017a.';
trainedClassifier.HowToPredict = sprintf('To make predictions on a new ...
table, T, use: \n  yfit = c.predictFcn(T) \nreplacing ''c'' with the name ... of
the variable that is this struct, e.g. ''trainedModel''. \n \nThe ... table,
T, must contain the variables returned by: \n          c.RequiredVariables ...
\nVariable formats (e.g. matrix/vector, datatype) must match the original ...
training data. \nAdditional variables are ignored. \n \nFor more ...
information, see <a href="matlab:helpview(fullfile(docroot, ''stats'', ...
''stats.map''), ''appclassification_exportmodeltoworkspace'')">How to ...
predict using an exported model</a>.');


% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = inputTable.Properties.VariableNames(1:largoTabla-1);
predictors = inputTable(:, predictorNames);
response = inputTable.objectType;
isCategoricalPredictor = [];
for k=1:largoTabla-1;
    isCategoricalPredictor(k) = 0;
end

% Perform cross-validation
partitionedModel = crossval(trainedClassifier.ClassificationSVM, 'KFold', 5);

% Compute validation predictions
[validationPredictions, validationScores] = kfoldPredict(partitionedModel);

% Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');
```

CAPÍTULO 10

Identificación de desgaste en la falda del pistón

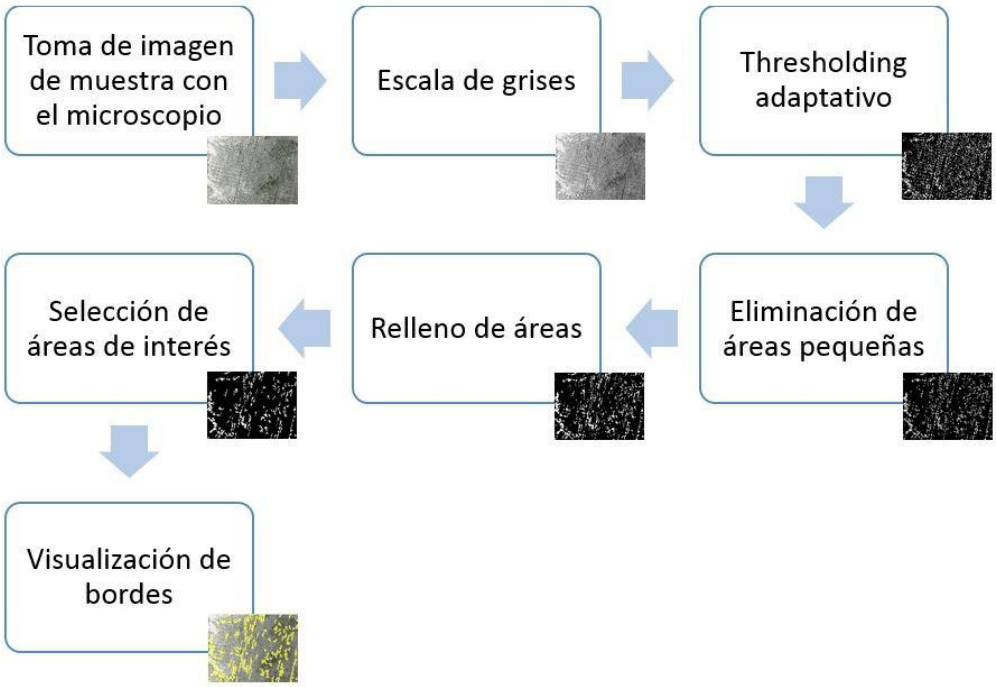


Figura 43: Etapas de la identificación de desgaste en la falda del pistón.

Se tomaron muestras de ocho distintas regiones de la falda del pistón, cuatro por cada sección. Luego de que el microscopio tomara las fotos, se sometieron a un pre-procesamiento para lograr seleccionar y resaltar las rayaduras debido al desgaste por suciedad. Se aplicó un Thresholding adaptativo con una sensibilidad de 0.45. Lo que se buscaba con este algoritmo era resaltar las imperfecciones más significativas en cada región de la falda, por lo que

fueron eliminadas las áreas con menos de 30 pixeles y se resaltaron las más grandes. Como resultado se obtuvo la imagen orignal junto con los bordes que encierran las imperfecciones más notables.

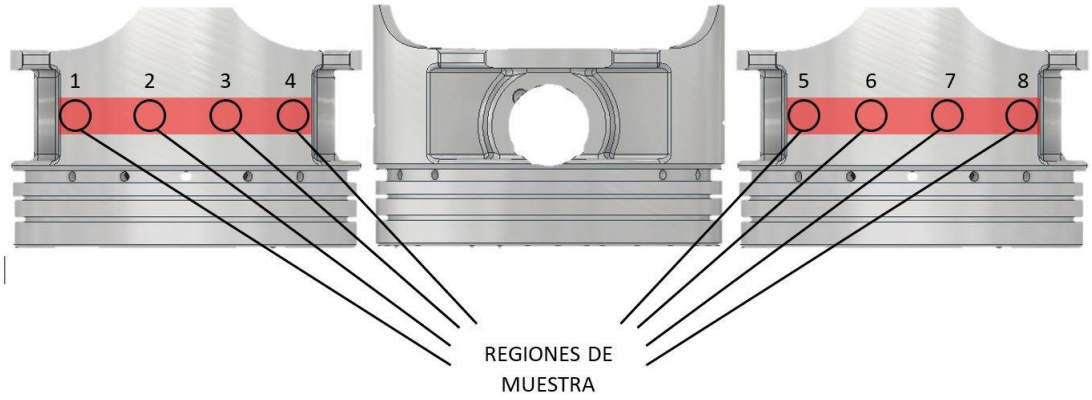


Figura 44: Regiones de muestra para la identificación de desgaste en la falda del pistón.

Cada imagen abarca un área de 5mmx5mm, tomadas a una altura igual al del bulón. El valor del área que abarca la imagen se tomó luego de haber calibrado el zoom del microscopio digital junto con la regla de calibración que trae el kit del dispositivo.

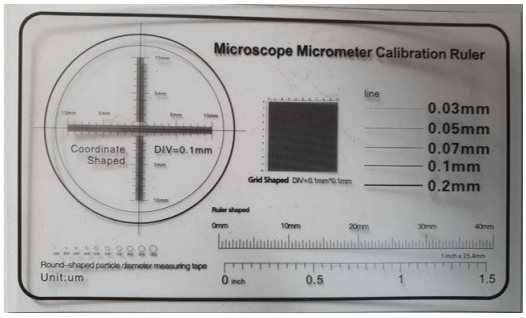


Figura 45: Regla de calibración del microscopio digital.

CAPÍTULO 11

Identificación y medición de las ranuras de los anillos

Para esta fase se ideó en medir las ranuras sin tocarlas con algún instrumento de medición, sino mediante sensores que utilizaran la proyección de una luz estructurada sobre la superficie de la sección de las ranuras. Se implementaron entonces dos alternativas con distintos métodos para calcular las medidas.

11.1. Mediante la cámara de escaneo de línea TSL1401CL

A través de la cámara se escaneaba una línea de la sección de las ranuras en forma perpendicular a la orientación de las mismas ranuras, obteniendo así una imagen unidimensional de 128 pixeles. Esta imagen unidimensional capta la intensidad de luz que reflejan las ranuras, cuando la intensidad es menor a cierto valor se interpreta como una sombra. A cada pixel se le asignó un tamaño en milímetros que varía en función del campo de visión (FOV) que capta la cámara. A su vez, el campo de visión varía en función de la distancia entre el lente y la superficie de las ranuras, ver ecuación 10, y del ángulo de visión.

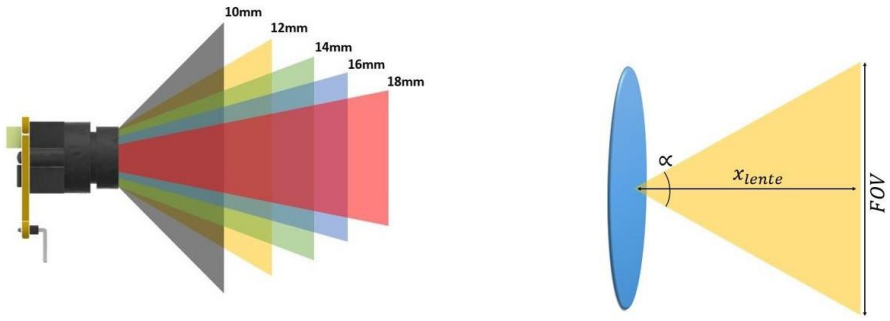


Figura 46: Relación del campo de visión, el ángulo de visión y la distancia entre el lente y el objeto.

La cámara seleccionada posee un lente ultra gran angular que permite variar la distancia focal en un rango de 10mm a 18mm. Se ajustó el lente a una distancia focal de 14mm, obteniendo un ángulo de visión de 114°[38]. Por lo tanto, para encontrar el tamaño de cada pixel se utiliza la siguiente ecuación, en donde Pc es la cantidad de pixeles de la cámara.

$$T_{pp} = \frac{FOV}{P_c} = \frac{2 \cdot x_{lente} \cdot \sin a}{P_c} \tag{12}$$

Al tener el tamaño por cada pixel, se procedió a implementar un algoritmo para el pre-procesamiento de la imagen unidimensional y lograr medir el ancho de cada ranura en función de la cantidad de pixeles obscuros. Estos pixeles obscuros se definieron mediante un valor de umbral, distinto para tipo de pistón ya que, debido al material del que están hechos, reflejaban la luz de distinta forma. Además, para facilitar el análisis, se seleccionó un rango de pixeles que abarcan solo las ranuras de los anillos dentro del campo de visión de la cámara.

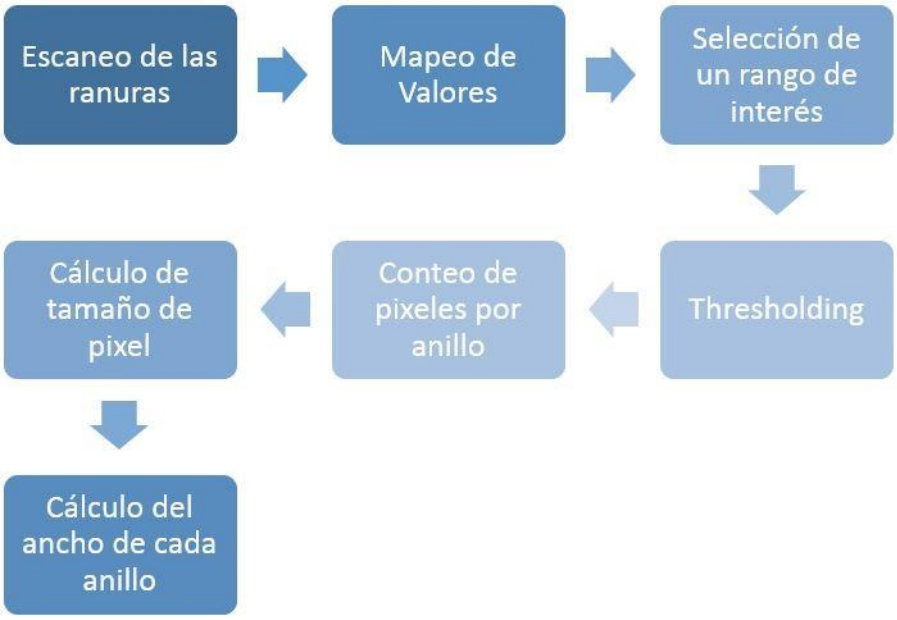


Figura 47: Proceso de medición de las ranuras con cámara de escaneo de línea.

11.2. Mediante barrido láser con el sensor GY-VL53L0X

El objetivo al emplear este sensor era obtener una nube de puntos que permitiera visualizar la forma geométrica en 2D de la sección de las ranuras de los anillos. Cada punto fue referenciado a la posición del láser dentro del área de trabajo. Debido a la profundidad de las ranuras, cada una de ellas se lograba identificar al observar un cambio significativo en las distancias de los puntos. Además, era necesario que el barrido contara con una resolución de por lo menos 0.01mm para lograr medir el ancho de cada ranura. Para esta aplicación se utilizó una resolución de 0.1mm obteniendo un total de 350 puntos.

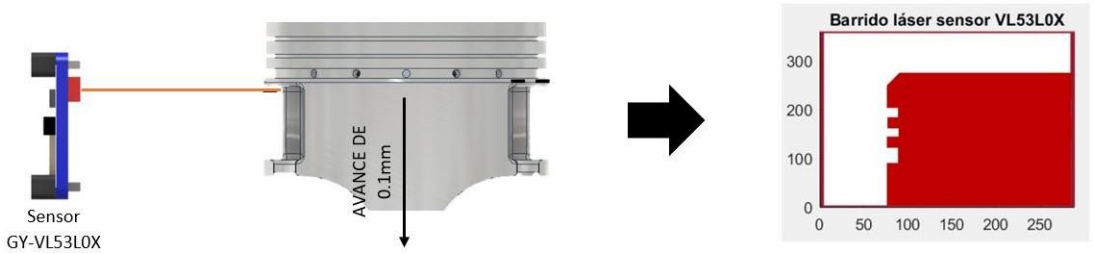


Figura 48: Escaneo 2D de la sección de las ranuras de un pistón con el sensor GY-VL53L0X.

CAPÍTULO 12

Desarrollo de la interfaz gráfica de usuario en MATLAB

Uno de los requerimientos del sistema es contar con una interfaz de usuario que permita un control sencillo del dispositivo electrónico y la visualización de los resultados del proceso de inspección y medición de los pistones. Se creó una ventana con distintos componentes gráficos dentro del entorno de programación visual de MATLAB, conocido como GUIDE. Entre los distintos componentes gráficos utilizados se encuentran: botones, cuadros de listas, etiquetas, cuadro de ejes, cajas de texto editables y paneles. La interfaz de usuario se comunica a través del puerto serial con el Arduino, transferiendo y recibiendo datos del microcontrolador.

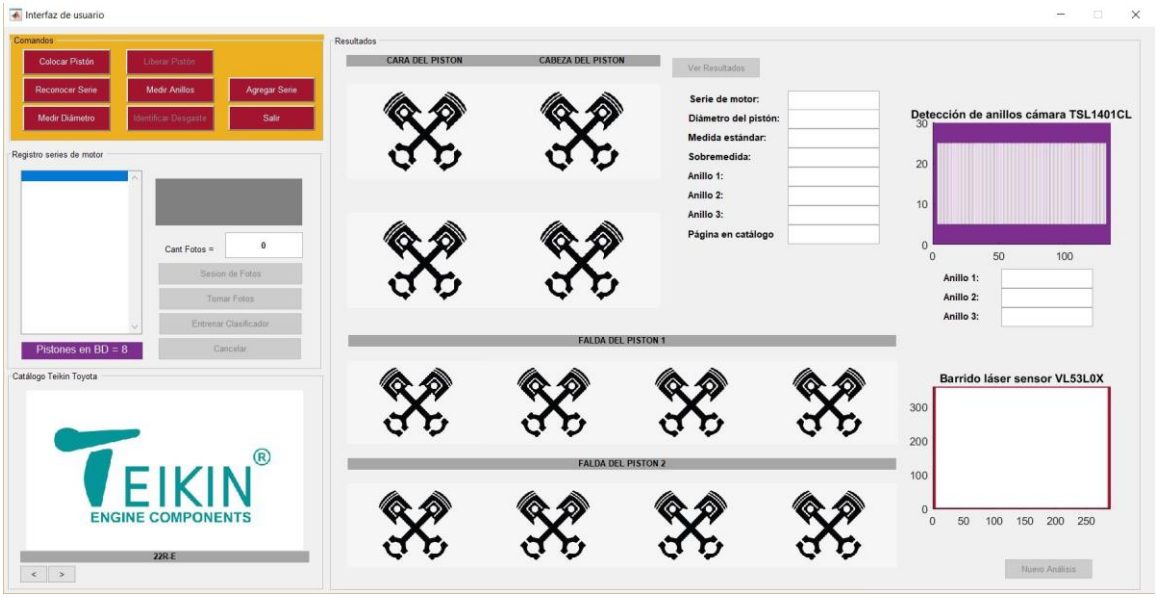


Figura 49: Interfaz de usuario para el despliegue de resultados y el registro de nuevos pistones.

Esta interfaz se encuentra dividida en cuatro paneles principales: comandos, registro de series de motor, catálogo Teikin Toyota y resultados.

12.1. Diseño gráftco de los paneles

12.1.1. Panel de comandos

Contiene todas las tareas que el dispositivo puede realizar, se incluye el proceso de inspección de los pistones, un comando para el registro de nuevas series de motor, dos comandos para sujetar el pistón en mecanismo de mandril y uno para establecer la conexión a través del puerto serial con el Arduino. Los botones se bloquean y desbloquean conforme se realizan las tareas, por lo que no es posible activar dos comandos a la vez.

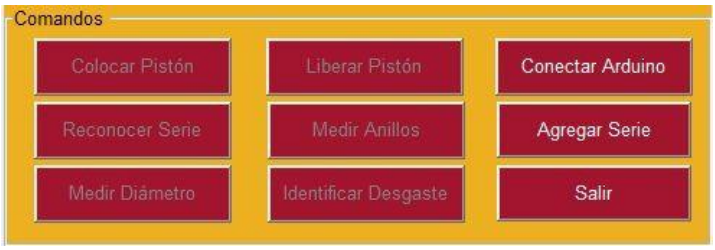


Figura 50: Panel de comandos de la interfaz de usuario.

12.1.2. Panel de registro de series de motor

Debido a que el método seleccionado para el reconocimiento de los pistones se encuentra limitado por la pequeña base de datos de 8 series de motor, se consideró añadir una función que permitiera expander la base de datos. En el lado izquierdo del panel se muestran todas las series de motor que aún faltan por registrar, conforme se van añadiendo la lista se actualiza. Cabe mencionar que si se desea registrar una nueva serie en la base de datos, es necesario contar con por lo menos 12 muestras. Cada vez que el dispositivo termina de tomar las 4 fotos de una muestra, permite al usuario colocar la siguiente muestra. Al finalizar se entrena un nuevo clasificador, mientras se sigan añadiendo más series, este proceso será cada vez más lento.

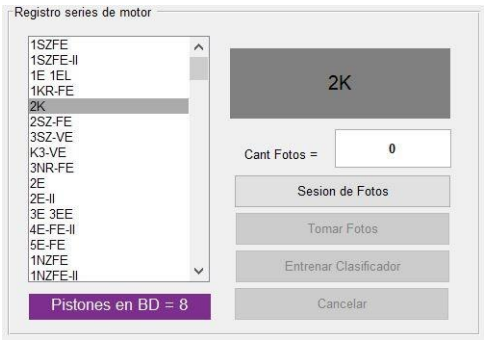


Figura 51: Panel de registro de series de motor de la interfaz de usuario.

12.1.3. Panel del catálogo Teikin Toyota

En este panel el usuario puede visualizar el catálogo Teikin de tal manera que solo puede visualizar las medidas estándar de un solo pistón a la vez. Se tienen dos botones que permiten al usuario navegar entre todos los pistones de la marca Toyota disponibles en el catálogo Teikin en su última edición.

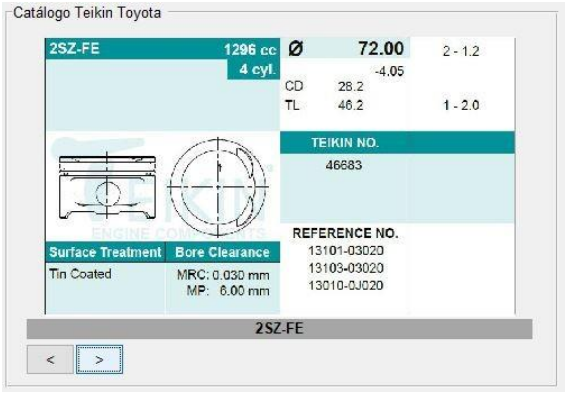


Figura 52: Panel de visualización del catálogo Teikin en la interfaz de usuario.

12.1.4. Panel de resultados

Se muestran todos los resultados del proceso de inspección de los pistones, presentando sus medidas estándar del catálogo Teikin. En la siguiente imagen se presentan los resultados de la inspección realizada a un piston perteneciente a la serie de motor 3VZE.

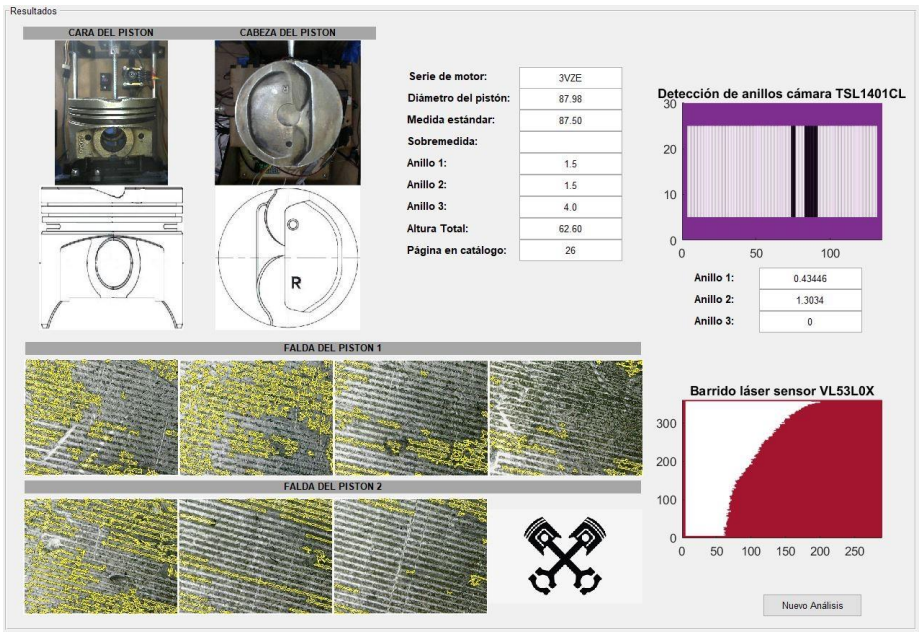


Figura 53: Panel de resultados en la interfaz de usuario.

12.2. Codiftcación de los componentes gráftcos

A cada botón se le codificó una función distinta que va relacionada con el proceso de inspección de los pistones. Se detallan a continuación los botones que se codificaron y el nombre que se les fue asignado dentro del GUIDE:

- Botón colocar pistón: pushbutton19
- Botón liberar pistón: pushbutton20
- Botón conectar Arduino: pushbutton21
- Botón reconocer serie: pushbutton3
- Botón medir anillos: pushbutton12
- Botón agregar serie: pushbutton7
- Botón medir diámetro: pushbutton13
- Botón identificar desgaste: pushbutton4
- Botón sesión de fotos: pushbutton8
- Botón tomar fotos: pushbutton9
- Botón entrenar clasificador: pushbutton10
- Botón cancelar: pushbutton11
- Botón atrás catálogo Teikin: pushbutton17
- Botón adelante catálogo Teikin: pushbutton18

La codificación de cada botón se detalla en la sección de Anexos.

CAPÍTULO 13

Resultados

13.1. Medición del diámetro del pistón

Se tomaron medidas del diámetro de 12 diferentes tipos de pistones con el prototipo 1 y el prototipo 2. Para verificar la exactitud del prototipo, los mismos pistones fueron medidos con un micrómetro. Al calcular el porcentaje de error puede notarse que los valores en el prototipo 1 están muy alejados de la medida real, esto se debió al pequeño pandeo que se producía en la base del mecanismo del comparador cada vez que el palpador tocaba al pistón. Además, el primer prototipo no tuvo un diseño preliminar en un software de modelado, por lo que las piezas fueron montadas con medidas aproximadas dentro del espacio de trabajo.

Cuadro 18: Medición del diámetro de pistones de distintas series de motor. Las medidas realizadas con el primer prototipo son comparadas con las medidas tomadas con un micrómetro convencional con un porcentaje de error

Serie de motor	Prototipo 1	Micrómetro convencional	Porcentaje de error
3E	67.20 mm	73.05 mm	8.01 %
3RZFE	90.50 mm	94.92 mm	4.66 %
3VZE	84.04 mm	87.45 mm	3.90 %
4AFE-II	76.28 mm	81.02 mm	5.85 %
4AFE-III	77.32 mm	81.23 mm	4.81 %
20R	82.61 mm	88.39 mm	6.54 %
22R	86.16 mm	92.05 mm	6.40 %
22R-E	86.56 mm	91.99 mm	5.90 %
1ZZFE	75.70 mm	78.96 mm	4.13 %
2AZFE	85.02 mm	88.49 mm	3.92 %
3A	74.58 mm	77.52 mm	3.79 %
21R	78.20 mm	83.97 mm	6.87 %

Para el prototipo 2 se realizó un diseño preliminar en 3D en el software Autodesk Inventor, se establecieron las distancias entre cada una de las piezas y se consideraron en el cálculo del diámetro. Puede notarse en la siguiente tabla que los valores se acercan más a las medidas reales, obteniendo una mejor exactitud. Además, para resolver el problema del pandeo en la extensión del palpador, se instaló una pieza que guía el movimiento horizontal del palpador hacia el pistón.

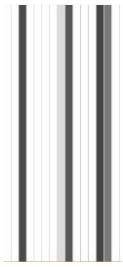
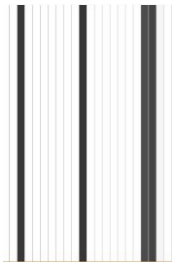
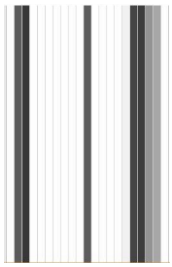
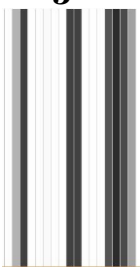
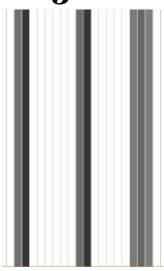
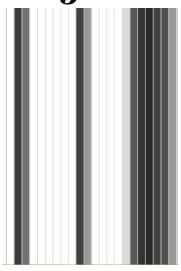
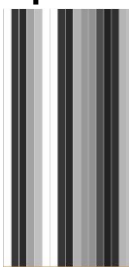
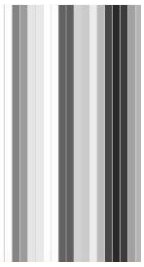
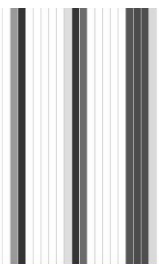
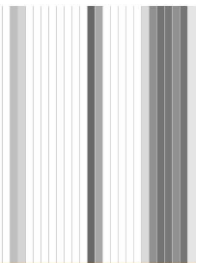
Cuadro 19: Medición del diámetro de pistones de distintas series de motor. Las medidas realizadas con el segundo prototipo son comparadas con las medidas tomadas con un micrómetro convencional con un porcentaje de error

Serie de motor	Prototipo 2	Micrómetro convencional	Porcentaje de error
3E	73.14 mm	73.05 mm	0.12 %
3RZFE	93.18 mm	94.92 mm	1.83 %
3VZE	86.52 mm	87.45 mm	1.06 %
4AFE-II	80.92 mm	81.02 mm	0.12 %
4AFE-III	80.78 mm	81.23 mm	0.55 %
20R	88.38 mm	88.39 mm	0.01 %
22R	91.74 mm	92.05 mm	0.34 %
22R-E	91.90 mm	91.99 mm	0.10 %
1ZZFE	78.82 mm	78.96 mm	0.18 %
2AZFE	88.02 mm	88.49 mm	0.53 %
3A	76.13 mm	77.52 mm	1.79 %
21R	83.48 mm	83.97 mm	0.58 %

13.2. Medición del ancho de las ranuras de los anillos

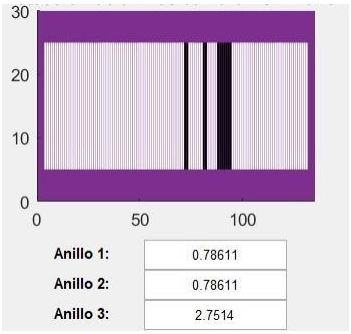
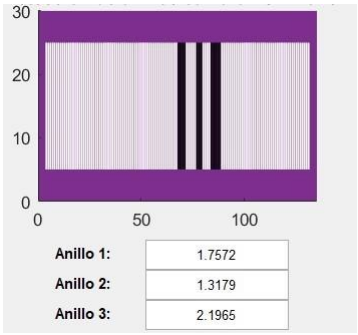
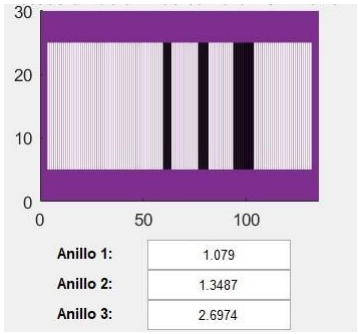
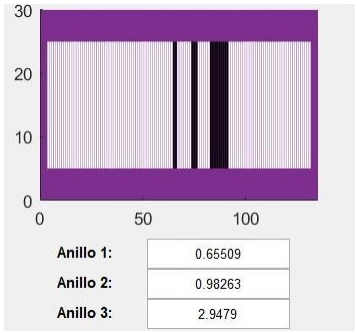
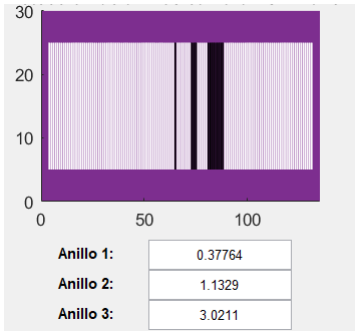
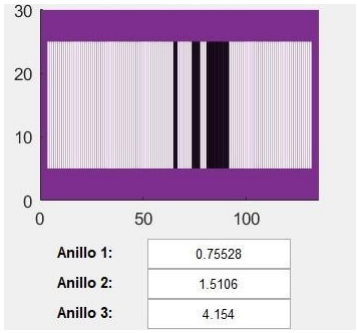
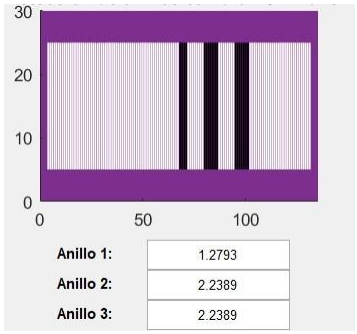
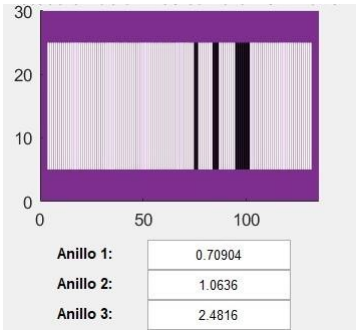
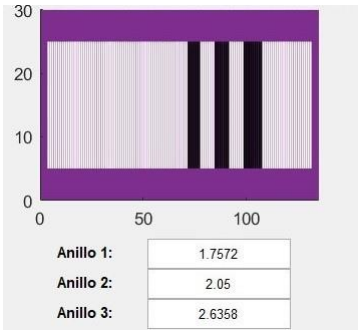
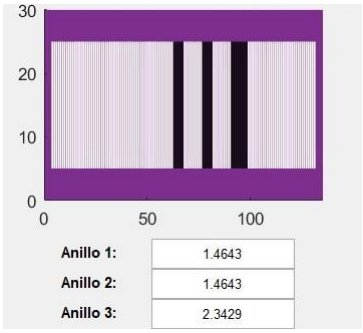
En el siguiente cuadro se pueden observar el escaneo de las ranuras de los anillos de 12 diferentes pistones que pertenecen a distintas series de motor. Estas imágenes se tomaron utilizando una linterna como fuente de luz para iluminar la región de las ranuras. Para el prototipo 1, el ancho fue calculado en base a criterio propio, no se implementó un algoritmo computarizado. Durante la fase del primer prototipo se realizaron numerosas pruebas para entender el funcionamiento de la cámara de escaneo de línea TSL1401CL.

Cuadro 20: Detección de las ranuras de los anillos con cámara TSL1401CL de distintas series de motor en el prototipo 1. Para cada imagen se describe la medida de cada anillo en milímetros.

1AZFE  Anillo 1 = 1 mm Anillo 2 = 1 mm Anillo 3 = 2 mm	**1ZZFE**  Anillo 1 = 1 mm Anillo 2 = 1 mm Anillo 3 = 2 mm	**2AZFE**  Anillo 1 = 1 mm Anillo 2 = 1 mm Anillo 3 = 2 mm	**2TZ**  Anillo 1 = 2 mm Anillo 2 = 1 mm Anillo 3 = 4 mm
3A  Anillo 1 = 1 mm Anillo 2 = 1 mm Anillo 3 = 3 mm	**3E**  Anillo 1 = 1 mm Anillo 2 = 2 mm Anillo 3 = 4 mm	**3RZ**  Anillo 1 = 2 mm Anillo 2 = 2 mm Anillo 3 = 3 mm	**3VZ**  Anillo 1 = 2 mm Anillo 2 = 2 mm Anillo 3 = 5 mm
4AFE  Anillo 1 = 2 mm Anillo 2 = 2 mm Anillo 3 = 3 mm	**22R**  Anillo 1 = 2 mm Anillo 2 = 2 mm Anillo 3 = 3 mm	**21RC**  Anillo 1 = 2 mm Anillo 2 = 2 mm Anillo 3 = 3 mm	**22R-E**  Anillo 1 = 2 mm Anillo 2 = 2 mm Anillo 3 = 4 mm

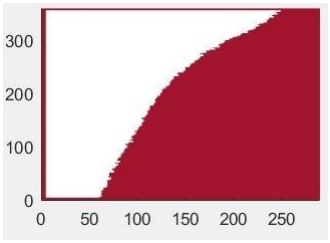
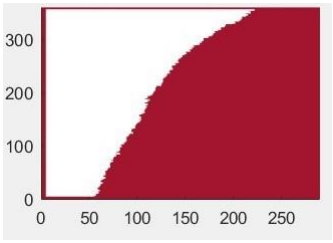
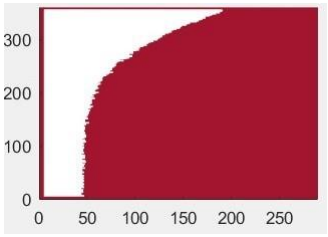
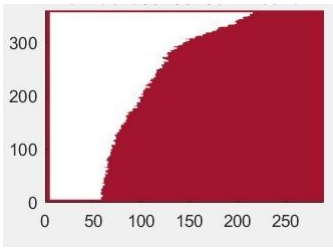
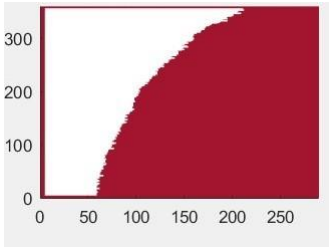
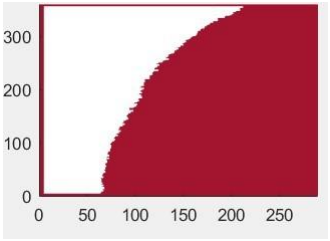
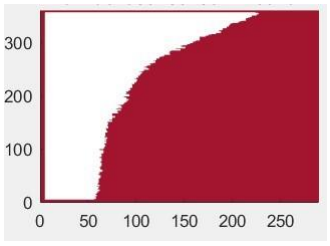
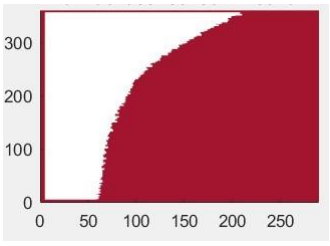
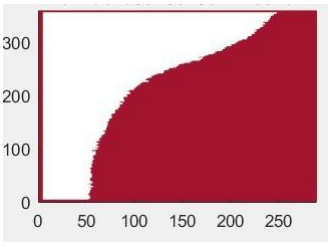
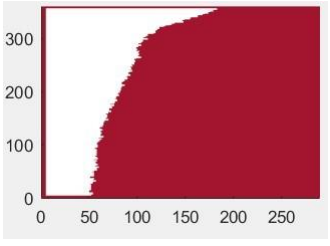
Para el prototipo final, se implementó el algoritmo para el cálculo del ancho de las ranuras en base a la cantidad de pixeles, tal y como se describe en el capítulo 11, obteniendo como resultados las imágenes que se observan en la siguiente tabla. A diferencia de la tabla anterior, solo se identificaron las ranuras de 10 series de motor porque dentro del algoritmo se aplicó un valor de umbral para estas series en específico.

Cuadro 21: Detección de las ranuras de los anillos con cámara TSL1401CL de distintas series de motor en el prototipo 2. Cada gráfico se extrajo de la interfaz de usuario implementado para el prototipo final.

1ZZFE 	**3E** 	**3RZFE** 
3VZE 	**4AFE-II** 	**4AFE-III** 
20R 	**21R** 	**22R** 
22R-E 		

Para el caso del sensor láser no fue posible escanear la geometría 2D de las ranuras a través de una nube de puntos. En la siguiente tabla se puede observar una curva que representa un cambio en la distancia entre la superficie del pistón y la superficie de fondo dentro del espacio de trabajo. Esto se debió a que el diámetro del láser emisor del sensor VL53L0X es de 0.4 mm (ver hoja de datos) y se estableció una resolución de 0.1 mm para la nube de puntos.

Cuadro 22: Escaneo láser de las ranuras de los anillos mediante el sensor VL53L0X en el prototipo 2. Cada gráfico se extrajo de la interfaz de usuario implementado para el prototipo final.

1ZZFE 	**3E** 	**3RZFE** 
3VZE 	**4AFE-II** 	**4AFE-III** 
20R 	**21R** 	**22R** 
22R-E 		

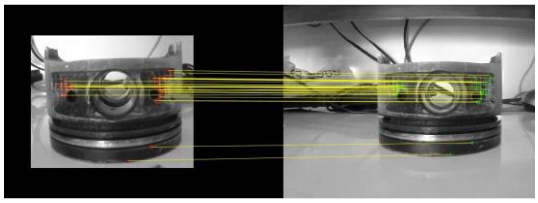
13.3. Fase de reconocimiento y clasiftcación de los pistones

13.3.1. Mediante Feature Points en Matlab

Al aplicar la detección de los puntos característicos más significativos en la cara de cada pistón de la base de datos, se compararon con pistones de la misma serie de motor pero en distintos entornos e iluminación. Tal y como puede observarse en las siguientes imágenes, la detección se logró en la mayoría de pistones. Sin embargo, con el pistón de la serie 22R no se obtuvieron suficientes *match points*.

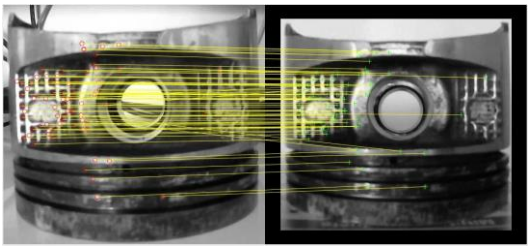
Cuadro 23: Reconocimiento de pistones mediante Feature Points.

22R-E

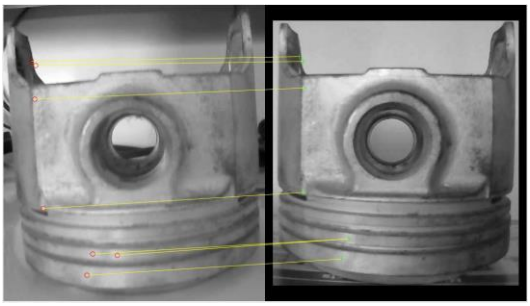


2AZFE

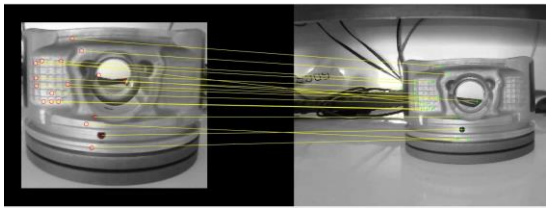
4AFE



22R

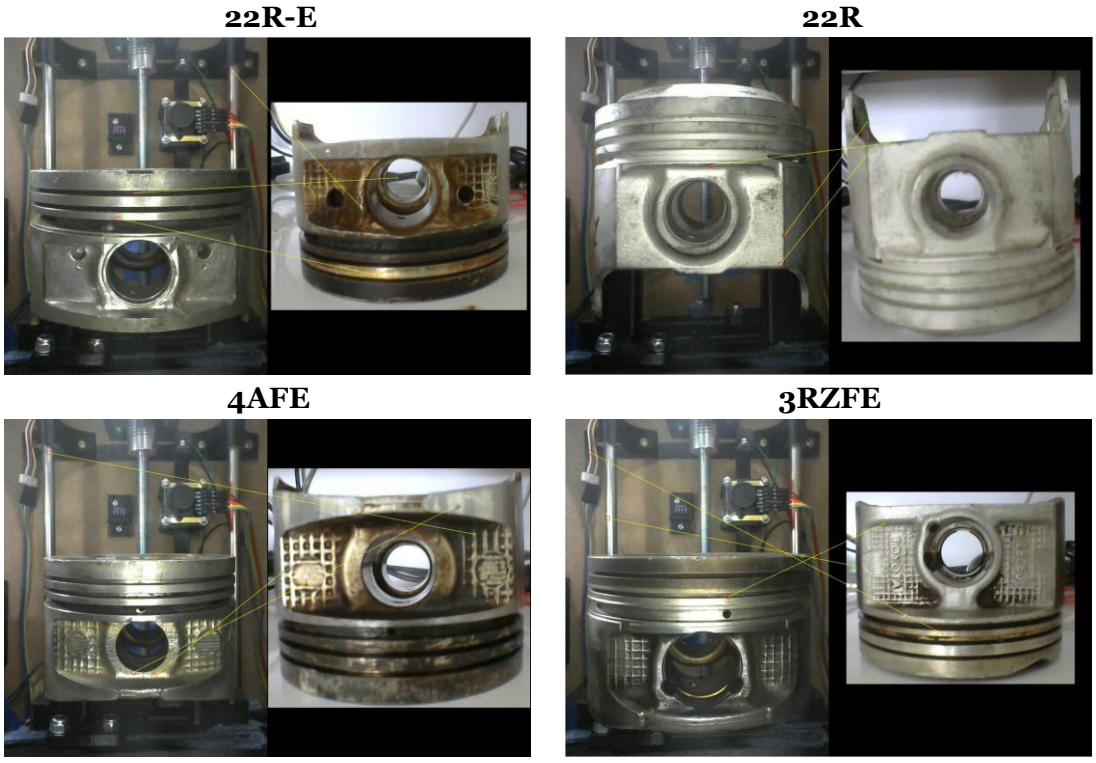


3RZFE



Sin embargo, estas pruebas se realizaron para una misma muestra de cada seri y cuando se implementó en el prototipo 2, no se obtuvieron los mismos resultados. Se utilizaron diferentes muestras y el entorno también era distinto.

Cuadro 24: Reconocimiento de pistones mediante Feature Points en el prototipo 2.



13.3.2. Mediante Machine Learning con Matlab

Debido a que las técnicas de procesamiento de imágenes aplicadas en las secciones anteriores no eran suficientes, se procedió a implementar algoritmos para el aprendizaje supervisado y creación de clasificadores. A través de la Classification Learner App se encontraron dos modelos lineales que representan los clasificadores de la cara y de la cabeza del pistón. Utilizando las máquinas de vectores, se decidió por utilizar el modelo lineal con un porcentaje de exactitud del 100 %. Los datos de entrenamiento se extrajeron de una base de datos de 8 series de motor distintas con 48 imágenes de cada serie, 24 imágenes de la cara y 24 imágenes de la cabeza, haciendo un total de 192 imágenes por clasificador y 12 pistones de muestra de cada serie. Cabe mencionar que cada pistón de muestra pasó por un proceso de limpieza antes de la toma de imágenes.

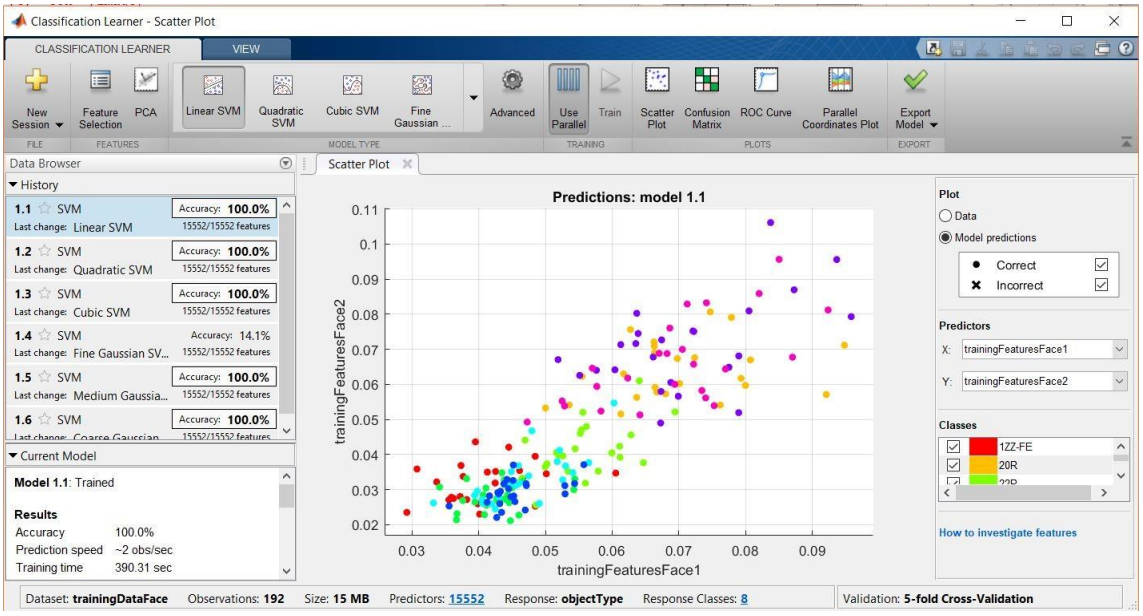


Figura 54: Entrenamiento de clasificador para la cara de los pistones. Entrenamiento para ocho series de pistones.

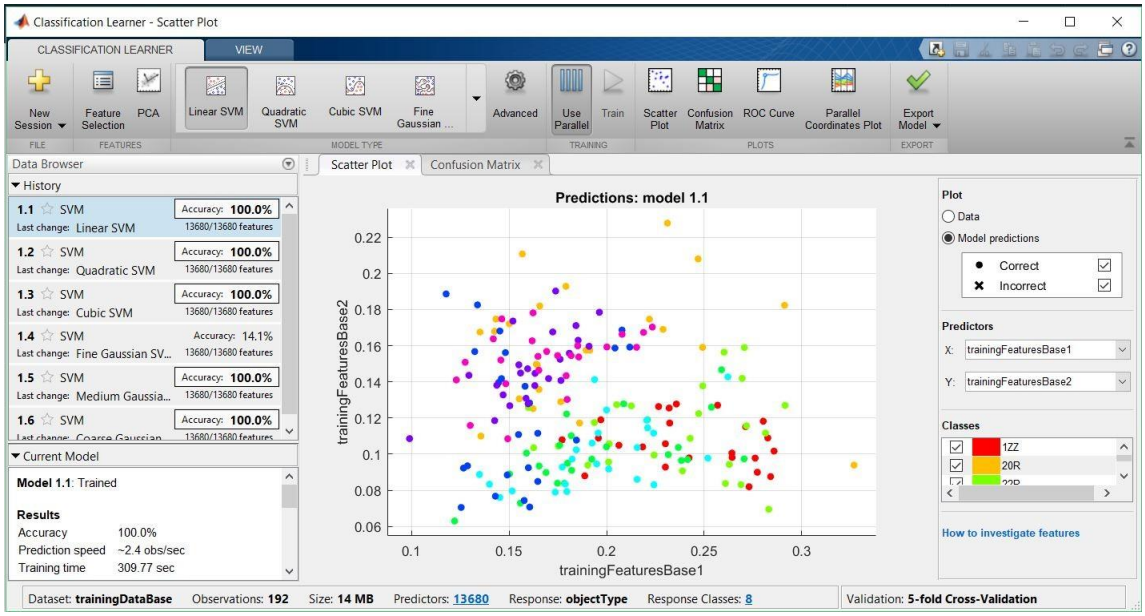


Figura 55: Entrenamiento de clasificador para la cabeza de los pistones. Entrenamiento para ocho series de pistones.

Las siguientes matrices de confusión representan la exactitud de cada modelo para las imágenes de los pistones de muestra de 8 distintas series de motor. Todos los pistones de muestra se recolectaron de una pila de pistones alojada dentro de una bodega de la empresa Reconstructora de Motores El Esfuerzo. No se registraron todas las series de motor gasolina de la marca Toyota debido a que no habían suficientes muestras para crear los datos de entrenamiento. Sin embargo, dentro de la interfaz gráfica se añadió una función que permite

expandir la base de datos mediante el registro de nuevas series de motor.

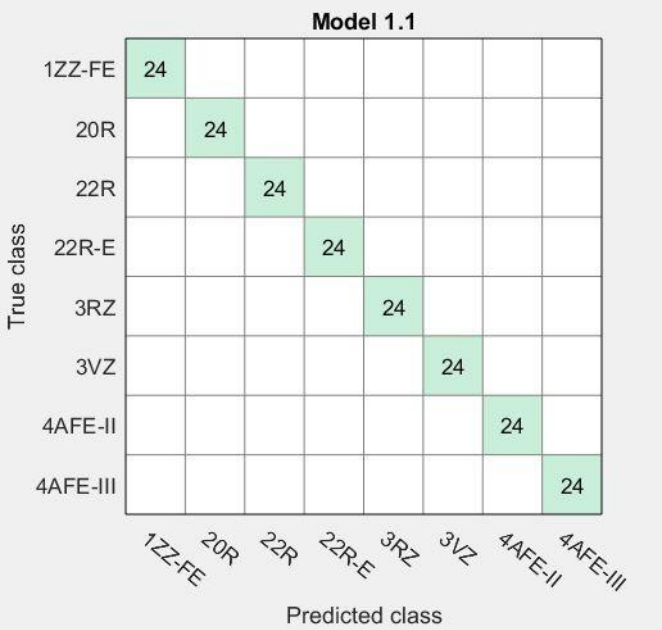


Figura 56: Matriz de confusión modelo de regresión Lineal para la cara de los pistones. Clasificador para ocho series de pistones.

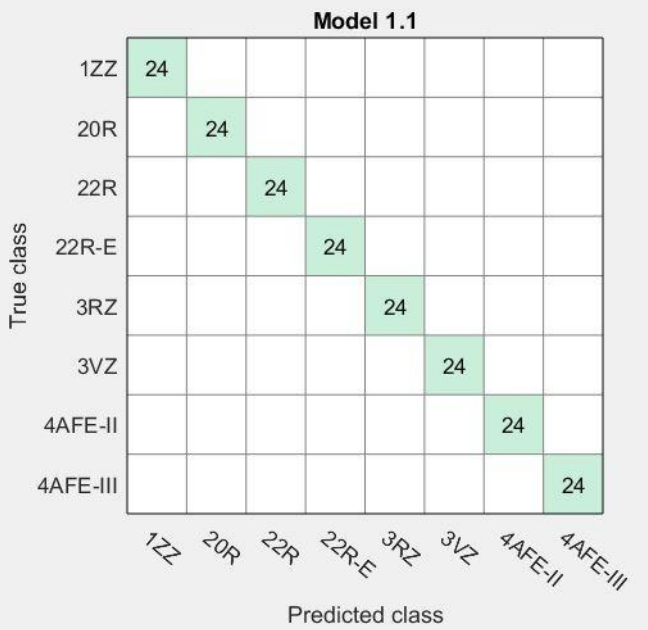


Figura 57: Matriz de confusión modelo de regresión Lineal para la cabeza de los pistones. Clasificador para ocho series de pistones.

Al implementarse los clasificadores entrenados dentro del área de trabajo del prototipo final se obtuvieron los resultados deseados, los clasificadores identificaron las ocho series

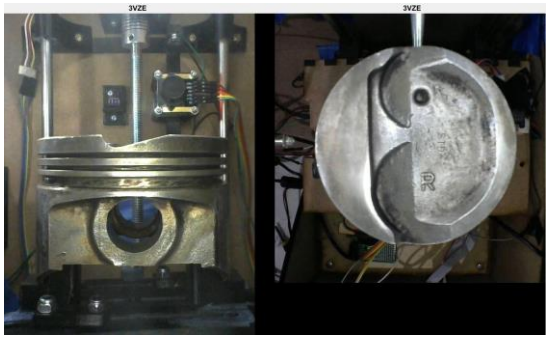
de motor sin problema. El inconveniente que se tiene es que la luz y el entorno donde se encuentran los pistones no debe modificarse, deben permanecer constantes.

Cuadro 25: Reconocimiento de la cara de pistones mediante Machine Learning.

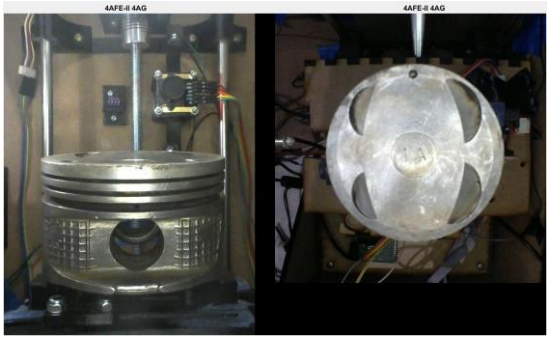
1ZZFE

3RZFE

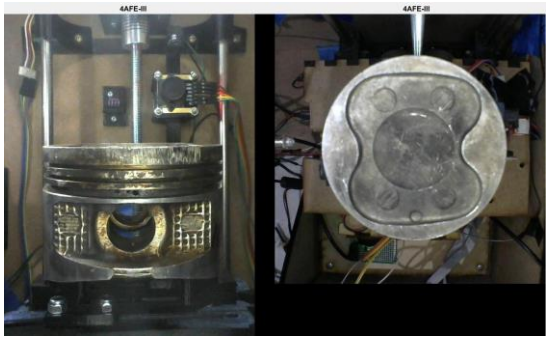
3VZE



4AFE-II



4AFE-III



20R

22R

22R-E

CAPÍTULO 14

Conclusiones

- Se logró implementar un mecanismo que permite la medición de los diámetros de un pistón con un porcentaje de error de 1.09 %
- Se logró implementar una cámara de detección de línea para detectar y medir el ancho de las ranuras de los anillos, como puede verse en el Cuadro 21
- El mecanismo de un mandril de dos mordazas diseñado cumple con los requerimientos para la medición del diámetro del pistón.
- Se implementó un algoritmo de Machine Learning para la clasificación de ocho series de pistones: 1ZZFE, 20R, 22R, 22R-E, 3RZ, 3VZ, 4AFE-II y 4AFE-III.
- Se implementó una función que permite expander la base de datos y crear nuevos clasificadores con el algoritmo de Machine Learning en Matlab.
- No fue posible medir el desgaste en los anillos debido a que la cámara posee una resolución de 128 pixeles, lo cual limitó obtener una medida menor a 1 mm.
- Se creó una base de datos de imágenes sobre pistones de motor gasolina de vehículos Toyota de las series: 1ZZFE, 20R, 22R, 22R-E, 3RZ, 3VZ, 4AFE-II y 4AFE-III.
- Se creó una interfaz gráfica en Matlab que permite la visualización de los resultados de las mediciones y toma de imágenes del pistón.

CAPÍTULO 15

Recomendaciones

- Ampliar la base de datos para que el algoritmo sea capaz de reconocer más series de pistones, no solo de Toyota, sino de otras marcas de motor.
- Reemplazar la cámara TSL1401CL por una cámara que se utilice en la industria para la inspección de productos manufacturados para obtener medidas con una resolución menor a 1 mm en la medición de la holgura de los anillos en sus respectivas ranuras del pistón.
- Considerar la implementación de la medición del diámetro del pistón con cortinas láser para futuras versiones mejoradas.
- Mejorar el mecanismo de sujeción 2-Jaw Chuck para lograr mediciones de los diámetros más precisas.
- Mejorar el mecanismo que permite girar el pistón sobre su propio eje para que logre tener un giro más suave y sólido.

CAPÍTULO 16

Bibliografía

[1] "Negocio automotriz en Centroamérica", Central America Data, inf. téc., oct. de 2018, https://www.centralamericadata.com/es/article/home/Negocio_automotriz_en_Centroamrica, visitado el 03-04-2019.

[2] *Measurement electronic gauge for piston control*, Infas, http://www.infas.com.ar/productos/medicion/motor/piston/dispositivo-electronico-de-medicion-y-control-de-piston/?lang=ent, visitado el 03-04-2019.

[3] *Multigauging Division*, MGPL Gauges for Age, http://mikronix-gauges.com/new/multigauging-division/, visitado el 03-04-2019, 2015.

[4] K. Anja, G. Peter y O. Ulrich. (2012). Line scan cameras, inspect internal O-rings groove surfaces. https://www.sukhamburg.com/download/photonik_intl_2013_01_028.pdf, visitado el 03-04-2019.

[5] *Application: Non-tactile piston measurement*, Mahr, https://www.mahr.com/en/Services/Production-metrology/The-news-and-practice-blog/?BlogContentID=20804&Blog_action=comment, visitado el 03-04-2019,2013.

[6] C. Simon, "Does the ring fit? The laser knows!", FRAMOS, inf. téc., oct. de 2014, https://www.opli.net/opli_magazine/imaging/2014/framos-provides-stihl-with-an-optical-test-method-to-ensure-quality-piston-production-greatly-enhancing-the-reliability-and-cost-efficiency-of-quality-control-oct-news/, visitado el 03-04-2019.

[7] *Inspection of piston rings at PSA*, Keyence, https://www.keyence.com/solutions/case-studies/psa.jsp, visitado el 03-04-2019.

[8] H. Joseph, *OpenCV computer vision with Python*. United Kingdom: Packt Publishing, 2013, ISBN: 9781782163930.

[9] S. Steve, *Igaging micrometers and calipers*, https://arduinotronics.blogspot.com/2015/09/igaging-micrometers-and-calipers.html, visitado el 03-04-2019, 2015.

[10] B. Walter, *Entretenimiento y reparación de motores de automóvil*. España: Editorial Reverte, 1979, ISBN: 9788429148084.

[11] S. A. Santiago, "Motores (Ciclos Formativos)", en. España: Editorial Editex, 2017, cap. 1-13, ISBN: 9788490031728.

[12] R. d. A. Antonio José y M. D. Marta, *Motores de combustión interna*. España, Madrid: Universidad Nacional de Educación a Distancia, 2015, ISBN: 9788436270860.

[13] *Manual de motor a gasolina Toyota – sistemas, inspección, averías y reparación*, Toyota, Mecánico automotriz, 2016.

[14] D. S. Esteban y F. R. Julián, *Mecanizado básico. Novedad 2017. Transporte y mantenimiento de vehículos*. España: Editex, 2017, ISBN: 9788491610465.

[15] *ABSOLUTE Digimatic Indicator ID-S Series 543-with Simple Design*, Mitutoyo, https://ecatalog.mitutoyo.com/ABSOLUTE-Digimatic-Indicator-ID-S-Series-543-with-Simple-Design-C1196.aspx, visitado el 25-03-2019.

[16] *Sensores de imagen lineal*, Toshiba, https://toshiba.semicon-storage.com/es/product/sensor/linear-sensor.html, visitado el 16-01-2019.

[17] G. M. Francisco Javier, *Videovigilancia: CCTV usando videos IP*. España: Editorial ELearning, S.L., 2011.

[18] C. R. Leonel, A. J. Griselda y C. M. Jesús, *Sensores y actuadores, aplicaciones con Arduino*. México, D.F.: Grupo Editorial Patria, 2014, ISBN: 9786074389364.

[19] K. Kye-Si y R. Steven, *Practical Guide to Machine Vision Software: An Introduction with LabVIEW*. United States of America: John Wiley & Sons, 2015, ISBN: 9783527337569.

[20] *Understanding Line Scan Camera Applications*, Teledyne DALSA, https:// www.inspect-online.com/file/track/7757/1, visitado el 25-03-2019, 2014.

[21] M. M. Julio, "Técnicas de visión por computador para la reconstrucción en tiempo real de la forma 3D de productos laminados", Tesis doct., Universidad de Oviedo, Asturias, España, 2008.

[22] M. José y P. Benjamín, "Procesamiento avanzado de imágenes digitales", Universidad de Salamanca, España, inf. téc., 2011.

[23] R. H. André Josué, "Visión por computador en una mesa de pruebas para la experimentación con micro-robots móviles en robótica de enjambre", 2018.

[24] D. Mery, "Visión por computador", Tesis doct., Universidad Católica de Chile, Santiago de Chile, 2004.

[25] B. Gary y K. Adrian, *Learning OpenCV: Computer vision with the OpenCV library*, 1.ª ed. United States of America: O'Reilly Media, Inc., 2008, ISBN: 9780596554040.

[26] G. Gabriel y J. Patreek, *OpenCV 3.x with Python by example*, 2.ª ed. United Kingdom: Packt Publishing, 2018, ISBN: 9781788396769.

[27] R. Adrian, *Image hashing with OpenCV and Python*, PyImage Search, https://www.pyimagesearch.com/2017/11/27/image-hashing-opencv-python/, visitado el 20-02-2019, 2017.

[28] K. Dr. Neal, *Looks like it*, http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html, visitado el 24-01-2019, 2011.

[29] P. Dmitry, *Wavelet image hash in Python*, https:// fullstackml.com/wavelet-image-hash-in-python-3504fdd282b5, visitado el 25-01-2019, 2016.

[30] *Object detection in computer vision*, MathWorks, https://www.mathworks.com/discovery/object-detection.html, visitado el 03-04-2019.

[31] *Machine Learning*, MathWorks, https://la.mathworks.com/discovery/machine-learning.html, visitado el 03-04-2019.

[32] *Feature detection and extraction*, MathWorks, https://www.mathworks.com/help/vision/feature-detection-and-extraction.html, visitado el 03-04-2019.

[33] *Feature extraction for compact representation of image data in computer vision*, MathWorks, https://www.mathworks.com/discovery/feature-extraction.html, visitado el 03-04-2019.

[34] G. Shengrong, L. Chunping y J. Yi, *Advanced Image and Video Processing Using MATLAB*. Japón: Springer, 2018, vol. 12, ISBN: 9783319772233.

[35] W. Toshikazu, H. Fay y L. Stephen, *Advances in Image and Video Technology: Third Pacific Rim Symposium*. Japón: Springer, 2009, ISBN: 9783540929574.

[36] *Técnica de Machine Learning para crear modelos predictivos a partir de datos de entrada y respuesta conocidos*, MathWorks, https://la.mathworks.com/discovery/supervised-learning.html#, visitado el 03-04-2019.

[37] C. O. Víctor, *Física: principios con aplicaciones*. España: Pearson Educacion, 2006, ISBN: 9789702606956.

[38] P. Joaquín, C. Luis y O. Jaime, *La imagen fotográfica*. España: AKAL Bellas Artes, 2007, vol. 3, ISBN: 9788446020004.

[39] I. P. S. de Estudios Gallegos, *Cuadernos de estudios gallegos*, v. 57. Instituto Padre Sarmiento de Estudios Gallegos, 2010. dirección: https://books.google.com.gt/books?id=uYpQ9NAt-nUC.

[40] *Measuring Outer Diameter*, Keyence, https://www.keyence.com/ss/products/measure/measurement_library/measuring/out_diameter/, visitado el 03-04-2019.

[41] *Customizable Vision System*, *Applications*, Keyence, https://www.keyence.com/products/vision/vision-sys/xg-7000/applications/application-02.jsp, visitado el 03-04-2019.

[42] *Teikin catalog - Automotive Toyota*, 18.ª ed., http://www.teikin.com/catalogue, visitado el 03-04-2019, Teikin, 2018.

CAPÍTULO 17

Anexos

17.1. Base de datos de pistones de motor gasolina de la marca Toyota

Cuadro 26: Series de pistones que identifica el clasificador. Cabeza del pistón

Cuadro 27: Series de pistones que identifica el clasificador. Cara del pistón

17.2. Prototipos funcionales

Figura 58: Primer prototipo funcional.

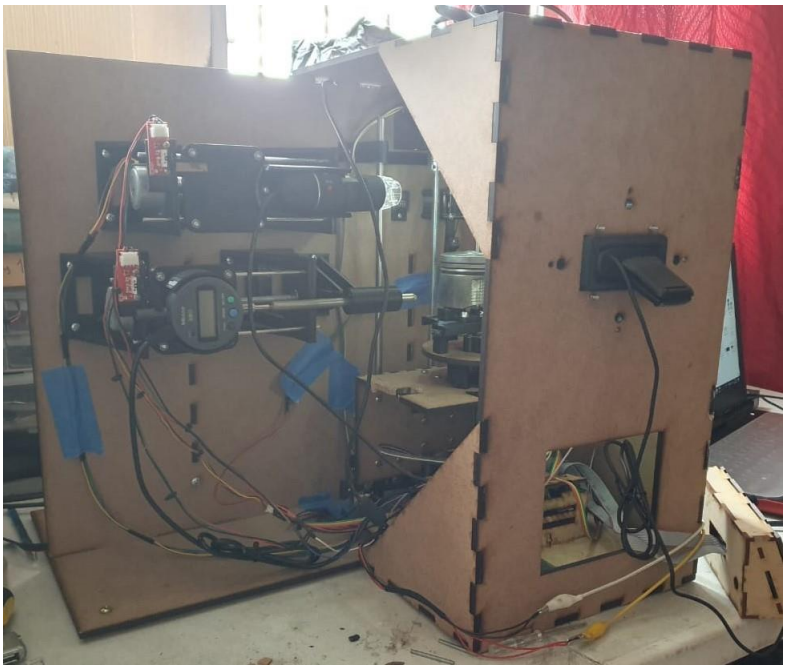


Figura 59: Segundo prototipo funcional.

17.3. Codiftcación de la respuesta de cada componente de la interfaz de usuario

17.3.1. Inicialización de la ventana de usuario

```
% --- Se ejecuta justo antes de que la ventana graficos sea visible
function graficos_OpeningFcn(hObject, eventdata, handles, varargin)
%-----------------------------------------------------------------------
%COMANDOS INTERNOS DE LA VENTANA
%-----------------------------------------------------------------------
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

%-----------------------------------------------------------------------
%DEFINICIÓN DE VARIABLES GLOBALES
%-----------------------------------------------------------------------
%Puerto comunicación serial
%-----------------------------------------------------------------------
global puerto;

puerto = serial('COM20','BaudRate',9600,'TimeOut',10,'Terminator','LF');
%Abriendo puerto para la comunicación
fopen(puerto);
```

```
%Área de recorte de las imágenes del pistón
%-------------------------------------------------------------------------
global recorteCara;
global recorteBase;

recorteCara = [275 150 625 800];
recorteBase = [170 50 675 720];

%Medidas estándar tomadas del catálogo Teikin
%-------------------------------------------------------------------------
global pistonesToyota;
global diametrosToyota;
global alturasToyota;
global  anillos1;
global anillos2;
global  anillos3;

%Nombre de las series de motor gasolina de los pistones marca Toyota
pistonesToyota  =  ["1SZFE"  "1SZFE-II"  "1E  1EL"  "1KR-FE"  "2K"
                    "2SZ-FE"  "3SZ-VE"  "K3-VE"  "3NR-FE"  "2E"
                    "2E-II" "3E 3EE" "4E-FE-II" "5E-FE" "1NZFE"
                    "1NZFE-II"  "1NZFE-III"  "1NZ-FXE"  "2NZ-FE"  "2NZ-FE-II"
                    "3K"  "4K  4KU"  "1GE"  "1G-EU"  "1G-FE"
                    "2A"  "2A-II"  "3P"  "4P(High)"  "4P(Low)"
                    "1A" "3A 3AU" "2R" "5AF" "5AFE NEW"
                    "1ZZFE" "3ZZ-FE" "4ZZFE" "12R  12RU"
                    "1S" "1ZR-FE" "2ZR-FE" "3ZR-FE" "3ZR-FE-II"
                    "5K"  "5K-II"  "7K"  "7K-II"  "4A-GEU"
                    "4A" "4AC,4AELU" "4AFE 4AG" "4AFE-II 4AG" "4AFE-III"
                    "4AGE"  "4AGE  4AGELC"  "4AGEC  4AGEU"  "4A-GEU  VVT"  "4A-GEU-II"
                    "7AFE" "2ZZFE" "4SFI" "4SFI-II" "5M"
                    "5MGEU" "7MGE 7MG" "7MGTE 7MGTEU" "21R 21RC"
                    "21R-II"  "21R-III"  "2S"  "13T"  "2T  2T-B"
                    "2T  2T-U"  "2T,2TB"  "2TC"  "3T3T-U13T-U"  "1AZFE"
                    "1AZ-FSE"  "1RZ"  "1RZ-III"  "1TR-FE"  "1Y"
                    "2Y" "3SFE" "3SFE-II" "3SFE-III" "3SFE-IV"
                    "3SFE-VII" "3S-FSE" "3SGE" "3SGE-II" "3SGTE"
                    "3Y" "2JZGE" "2JZGE-II" "2JZGTE" "5SFE"
                    "5SFE-II"  "1MZFE"  "2VZ-FE"  "3GR-FE"  "3VZE"
                    "3VZ-FE"  "1UZFE"  "1UZFE-II"  "5R  NEW"  "5R  OLD"
                    "18R  18RC  18RB"  "20R"  "20R-II"  "2AZFE"  "2AZFE-II"
                    "1AR-FE"  "F  FA-90"  "4Y"  "22R"  "22R-E"
                    "3MZFE"  "5VZ-FE"  "1GR-FE"  "2F"  "2F-II"
                    "2GR-FE"  "3F"  "3F  NEW  3FE"  "2UZFE"  "3UR-FE"
                    "2RZ 2RZE" "2RZ-FE" "2RZ-FE-II" "2TR-FE" "2TZ-FE"
                    "2TZ-FE-II"  "2TZFZE"  "3RZ-F"  "3RZ-FE  3RZ-FZ"
                    "1FZ-FE" "1FZ-FE-II" "1FZ-FE-III"];
```

```
%Diámetros en mm  de los pistones marca Toyota
diametrosToyota = ["69.00" "69.00" "70.50" "71.00" "72.00" "72.00" "72.00"
                   "72.00" "72.50" "73.00" "73.00" "73.00" "74.00" "74.00"
                   "75.00" "75.00" "75.00" "75.00" "75.00" "75.00" "75.00"
                   "75.00" "75.00" "75.00" "75.00" "76.00" "76.00" "76.60"
                   "76.60" "76.60" "77.50" "77.50" "78.00" "78.70" "78.70"
                   "79.00" "79.00" "79.00" "80.50" "80.50" "80.50" "80.50"
                   "80.50" "80.50" "80.50" "80.50" "80.50" "80.50" "81.00"
                   "81.00" "81.00" "81.00" "81.00" "81.00" "81.00" "81.00"
                   "81.00" "81.00" "81.00" "81.00" "82.00" "82.50" "82.50"
                   "83.00" "83.00" "83.00" "83.00" "84.00" "84.00" "84.00"
                   "84.00" "85.00" "85.00" "85.00" "85.00" "85.00" "85.00"
                   "86.00" "86.00" "86.00" "86.00" "86.00" "86.00" "86.00"
                   "86.00" "86.00" "86.00" "86.00" "86.00" "86.00" "86.00"
                   "86.00" "86.00" "86.00" "86.00" "86.00" "86.00" "87.00"
                   "87.00" "87.50" "87.50" "87.50" "87.50" "87.50" "87.50"
                   "87.50" "88.00" "88.00" "88.50" "88.50" "88.50" "88.50"
                   "88.50" "90.00" "90.00" "91.00" "92.00" "92.00" "92.00"
                   "93.50" "94.00" "94.00" "94.00" "94.00" "94.00" "94.00"
                   "94.00" "94.00" "95.00" "95.00" "95.00" "95.00" "95.00"
                  "95.00" "95.00" "95.00" "95.00" "100.00" "100.00" "100.00"];

%Altura en mm  de los pistones marca Toyota
alturasToyota = ["44.60" "44.40" "56.00" "47.00" "72.10" "46.20" "45.50"
                 "45.50" "49.80" "58.00" "58.00" "62.00" "62.80" "58.90"
                 "48.80" "49.80" "47.70" "47.90" "47.20" "47.20" "74.80"
                 "68.90" "67.60" "62.50" "52.80" "65.00" "64.30" "76.00"
                 "71.00" "72.40" "61.00" "60.90" "78.00" "59.50" "55.60"
                 "51.70" "52.20" "51.70" "79.90" "68.00" "55.00" "50.10"
                 "51.20" "51.30" "69.00" "69.00" "59.00" "58.70" "66.00"
                 "60.50" "60.60" "55.80" "55.80" "61.00" "64.60" "64.70"
                 "61.00" "50.20" "65.00" "55.60" "57.70" "70.20" "68.00"
                 "71.00" "72.50" "70.90" "70.90" "84.00" "65.50" "84.10"
                 "70.30" "97.30" "90.60" "91.80" "91.80" "99.40" "82.40"
                 "61.00" "62.00" "67.40" "59.00" "55.00" "74.00" "70.10"
                 "68.40" "67.10" "67.90" "65.50" "59.20" "66.60" "69.20"
                 "71.60" "62.10" "70.10" "65.00" "59.00" "65.00" "62.90"
                 "56.50" "51.30" "65.00" "54.60" "62.60" "62.20" "65.00"
                 "55.80" "85.00" "85.00" "84.90" "88.70" "88.70" "55.80"
                 "55.80" "53.00" "95.50" "65.00" "92.50" "63.90" "54.50"
                "63.00" "56.50" "95.80" "102.30" "52.50" "79.70" "79.70"
                "60.30" "55.30" "67.60" "58.90" "59.00" "59.40" "67.40"
                "67.00" "58.00" "61.00" "61.00" "76.00" "76.00" "66.00"];

%Ancho en mm de la ranura 1 de los pistones marca Toyota
anillos1 = ["1.2" "1.2" "1.5" "1.0" "2.0" "1.2" "1.2" "1.2" "1.0" "1.5"
            "1.5" "1.5" "1.2" "1.2" "1.2" "1.2" "1.0" "1.0" "1.2" "1.2"
            "2.0" "1.5" "1.5" "1.5" "1.2" "2.0" "2.0" "2.0" "2.0" "2.0"
```

```
                "1.5" "1.5" "2.0" "1.5" "1.2" "1.2" "1.2" "1.2" "2.0" "1.5"
                "1.0" "1.0" "1.0" "1.2" "1.5" "1.5" "1.5" "1.5" "1.2" "1.5"
                "1.5" "1.2" "1.2" "1.5" "1.5" "1.5" "1.5" "1.2" "1.5" "1.2"
                "1.2" "1.5" "1.5" "2.0" "1.5" "1.5" "1.5" "1.5" "1.5" "2.0"
                "1.5" "1.5" "2.0" "1.5" "2.0" "2.0" "1.5" "1.2" "1.2" "1.75"
                "1.75" "1.2" "1.5" "1.5" "1.5" "1.2" "1.5" "1.5" "1.2" "1.2"
                "1.2" "1.2" "1.2" "1.5" "1.5" "1.5" "1.5" "1.5" "1.2" "1.2"
                "1.5" "1.2" "1.5" "1.5" "1.5" "1.2" "2.0" "2.0" "2.0" "2.0"
                "2.0" "1.2" "1.0" "1.0" "3.0" "1.5" "2.0" "1.5" "1.2" "1.5"
                "1.2" "2.0" "2.0" "1.2" "1.5" "2.0" "1.5" "1.2" "1.75" "1.5"
                "1.75" "1.2" "1.75" "1.75" "1.75" "1.75" "1.5" "2.0" "1.75" "1.75"];

%Ancho en mm  de la ranura 2 de los pistones marca Toyota
anillos2 = ["1.2"  "1.2"  "1.5"  "1.0"  "2.0"  "1.2"  "1.2"  "1.2"  "1.0"  "1.5"
            "1.5" "1.5" "1.2" "1.2" "1.2" "1.2" "1.2" "1.2" "1.2" "1.2"
            "2.0" "1.5" "1.5" "1.5" "1.2" "2.0" "2.0" "2.0" "2.0" "2.0"
            "1.5" "1.5" "2.5" "1.5" "1.5" "1.2" "1.2" "1.2" "2.0" "1.5"
            "1.0" "1.0" "1.0" "1.0" "1.5" "1.5" "1.5" "1.5" "1.5" "1.5"
            "1.5" "1.5" "1.5" "1.5" "1.5" "1.5" "1.5" "1.5" "1.5" "1.5"
            "1.2" "1.5" "1.5" "2.0" "1.5" "1.5" "1.5" "2.0" "1.5" "2.0"
            "1.5" "1.5" "2.0" "1.5" "2.0" "2.0" "1.5" "1.2" "1.2" "1.5"
            "1.5" "1.2" "1.5" "1.5" "1.5" "1.5" "1.5" "1.5" "1.2" "1.2"
            "1.5" "1.5" "1.5" "1.5" "1.5" "1.5" "1.5" "1.5" "1.2" "1.2"
            "1.5" "1.2" "1.5" "1.5" "1.5" "1.2" "2.5" "2.5" "2.5" "2.5"
            "2.5" "1.2" "1.0" "1.0" "3.0" "1.5" "2.5" "1.5" "1.2" "1.5"
            "1.2" "2.5" "2.5" "1.2" "1.5" "2.0" "1.5" "1.2" "1.5" "1.5"
            "1.5" "1.2" "1.5" "1.5" "1.5" "1.5" "1.5" "2.0" "2.0" "2.0"];

%Ancho en mm  de la ranura 3 de los pistones marca Toyota
anillos3 = ["2.0"  "2.0"  "3.0"  "1.5"  "4.0"  "2.0"  "2.0"  "2.0"  "2.0" "3.0"
            "3.0" "3.0" "3.0" "3.0" "2.0" "2.0" "1.5" "2.0" "3.0" "2.0"
            "4.0" "4.0" "4.0" "4.0" "3.0" "4.0" "4.0" "4.0" "4.0" "4.0"
            "2.8" "2.8" "4.0" "3.0" "3.0" "3.0" "3.0" "3.0" "4.0" "4.0"
            "1.5" "1.5" "1.5" "2.5" "4.0" "4.0" "4.0" "4.0" "2.8" "2.8"
            "2.8" "3.0" "3.0" "3.0" "2.8" "2.8" "2.8" "3.0" "2.8" "3.0"
            "3.0" "4.0" "4.0" "4.0" "4.0" "4.0" "4.0" "4.0" "4.0" "4.0"
            "4.0" "4.0" "4.0" "4.0" "4.0" "4.0" "4.0" "2.0" "2.0" "4.0"
            "4.0" "2.0" "4.0" "4.0" "4.0" "4.0" "4.0" "4.0" "3.0" "2.0"
            "4.0" "4.0" "4.0" "4.0" "4.0" "4.0" "4.0" "4.0" "3.0" "3.0"
            "3.0" "2.0" "4.0" "4.0" "3.0" "3.0" "4.0" "4.0" "4.0" "4.0"
            "4.0" "2.0" "2.0" "2.0" "4.0" "4.0" "4.0" "4.0" "3.0" "4.0"
            "2.0" "4.0" "4.0" "2.0" "4.0" "4.0" "4.0" "2.0" "4.0" "4.0"
            "4.0" "2.0" "4.0" "4.0" "4.0" "4.0" "4.0" "4.0" "4.0" "4.0"];

%Cantidad de fotos tomadas en el registro de series
%-------------------------------------------------------------------------
global  cantFotos;
global numImages;
```

```
global numImagesPP;

cantFotos = 0;
numImages = 192;
numImagesPP = 24;

%Tamaño de la celda para los features
%-------------------------------------------------------------------------
global cellSize;

cellSize = [32 32];

%Modelos clasificadores entrenados
%-------------------------------------------------------------------------
global clasificadorCara;
global clasificadorBase;

%Se cargan los modelos desde el workspace de Matlab
clasificadorCara = evalin('base', 'clasificadorCara');
clasificadorBase = evalin('base', 'clasificadorBase');

%-------------------------------------------------------------------------
%VISUALIZACIÓN DE IMÁGENES DE INICIO EN LOS GRÁFICOS
%-------------------------------------------------------------------------
%Gráfico catálogo Teikin
%-------------------------------------------------------------------------
axes(handles.axes13);
imshow('teikinLogo.png');

%Gráficos cara del  pistón
%-------------------------------------------------------------------------
axes(handles.axes9);
imshow('pistonLogo.png');
axes(handles.axes10);
imshow('pistonLogo.png');

%Gráficos cabeza del pistón
%-------------------------------------------------------------------------
axes(handles.axes11);
imshow('pistonLogo.png');
axes(handles.axes12);
imshow('pistonLogo.png');

%Gráficos falda del pistón 1
%-------------------------------------------------------------------------
axes(handles.axes14);
imshow('pistonLogo.png');
axes(handles.axes15);
```

```
imshow('pistonLogo.png');
axes(handles.axes16);
imshow('pistonLogo.png');
axes(handles.axes17);
imshow('pistonLogo.png');

%Gráficos falda del pistón 2
%-------------------------------------------------------------------------
axes(handles.axes18);
imshow('pistonLogo.png');
axes(handles.axes19);
imshow('pistonLogo.png');
axes(handles.axes20);
imshow('pistonLogo.png');
axes(handles.axes21);
imshow('pistonLogo.png');

%-----------------------------------------------------------------
%INICIANDO GRÁFICO PARA EL SENSOR Y LA CÁMARA DE LÍNEA
%-------------------------------------------------------------------------
%Gráfico del sensor de distancia GY-VL53L0X
%-------------------------------------------------------------------------
axes(handles.axes6);
title('Barrido láser sensor VL53L0X');
xlim([0 290])
ylim([0 360])

%Ploteando 350 líneas horizontales con un ancho de 280
%de color blanco y grosor de 1
for i=5:355
    X = [5 285];
    Y = [i i];
    line(X,Y, 'Color', 'white', 'LineWidth', 1);
end

%Gráfico de la cámara de escaneo de línea TSL1401CL
%-------------------------------------------------------------------------
axes(handles.axes7);
title('Detección de anillos cámara TSL1401CL');
xlim([0 135])
ylim([0 30])

%Ploteando 128 líneas verticales con un ancho de 20
%de color variable entre 0-1 y grosor de 1
for i=4:131
    X = [i i];
    Y = [5 25];
    line(X,Y, 'Color', 'white', 'LineWidth', 1);
```

```
end

%--------------------------------------------------------------------
%EDITANDO TEXTO EN CAJA DE LISTA Y ETIQUETA
%--------------------------------------------------------------------
%Caja de lista de las series de motor por registrar
set(handles.listbox1, 'String', "");

%Etiqueta que indica la cantidad de fotos tomadas
set(handles.text11, 'String', "0");

%Etiqueta que indica la cantidad de series de motor
%registradas en la base de datos
cantPistones = numImages/numImagesPP;
set(handles.text13, 'String', "Pistones en BD = "+string(cantPistones));
```

17.3.2. Botón colocar pistón

```
% --- Se ejecuta al presionar el pushbutton19.
function pushbutton19_Callback(hObject, eventdata, handles)
%Uso de la variable global puerto para la comunicación
%--------------------------------------------------------------------
global puerto;

%Desactivando botones
%--------------------------------------------------------------------
set(handles.pushbutton19, 'Enable', 'off');  %Botón colocar pistón
set(handles.pushbutton5, 'Enable', 'off');   %Botón Salir
set(handles.pushbutton7, 'Enable', 'off');   %Botón Agregar serie

%Enviando datos a través del puerto
%--------------------------------------------------------------------
fwrite(puerto, 'f');
pause(3);

%Activando botones
%--------------------------------------------------------------------
set(handles.pushbutton3, 'Enable', 'on');    %Botón Reconocer serie
set(handles.pushbutton20, 'Enable', 'on');   %Botón Liberar pistón
set(handles.pushbutton5, 'Enable', 'on');    %Botón Salir
set(handles.pushbutton7, 'Enable', 'on');    %Botón Agregar serie
```

17.3.3. Botón liberar pistón

```
% --- Se ejecuta al presionar el pushbutton20
function pushbutton20_Callback(hObject, eventdata, handles)
```

```
%Uso de la variable global puerto para la comunicación
%--------------------------------------------------------------------------
global puerto;

%Desactivando botones
%--------------------------------------------------------------------------
set(handles.pushbutton20, 'Enable', 'off');  %Botón liberar pistón
set(handles.pushbutton3, 'Enable', 'off');   %Botón Reconocer serie
set(handles.pushbutton5, 'Enable', 'off');   %Botón Salir
set(handles.pushbutton7, 'Enable', 'off');   %Botón Agregar serie

%Enviando datos a través del puerto
%--------------------------------------------------------------------------
fwrite(puerto, 'g');
pause(3);

%Activando botones
%--------------------------------------------------------------------------
set(handles.pushbutton19, 'Enable', 'on');   %Botón colocar pistón
set(handles.pushbutton5, 'Enable', 'on');    %Botón Salir
set(handles.pushbutton7, 'Enable', 'on');    %Botón Agregar serie
```

17.3.4. Botón conectar Arduino

```
% --- Se ejecuta al presionar el pushbutton21
function pushbutton21_Callback(hObject, eventdata, handles)
%Uso de la variable global puerto para la comunicación
%--------------------------------------------------------------------------
global puerto;

%Desactivando el botón conectar Arduino
set(handles.pushbutton21, 'Enable', 'off');

%Enviando datos a través del puerto
fwrite(puerto, 'z');

%Activando el botón colocar pistón
set(handles.pushbutton19, 'Enable', 'on');
```

17.3.5. Botón reconocer serie

```
% --- Se ejecuta al presionar el pushbutton3
function pushbutton3_Callback(hObject, eventdata, handles)
%Uso de variables globales para el uso de las cámaras, comunicación con
%el puerto, los clasificadores entrenados, mediciones estándar, tamaño
%de celda, imágenes de muestra, predicción de la serie de motor
```

```
%-------------------------------------------------------------------------
global logitechFront;
global logitechUp;
global puerto;
global cellSize;
global clasificadorCara;
global clasificadorBase;
global recorteCara;
global recorteBase;
global caraTest;
global baseTest;
global prediccionSerie;
global pistonesToyota;
global alturasToyota;
global diametrosToyota;
global anillos1;
global anillos2;
global anillos3;

%Desactivando botones
%-------------------------------------------------------------------------
set(handles.pushbutton3, 'Enable', 'off');   %Botón reconocer serie
set(handles.pushbutton5, 'Enable', 'off');   %Botón salir
set(handles.pushbutton7, 'Enable', 'off');   %Botón agregar serie
set(handles.pushbutton20, 'Enable', 'off');  %Botón liberar pistón

%Proceso captura de imágenes de muestra
%-------------------------------------------------------------------------
%Enviando datos a través del puerto
fwrite(puerto, 'd');


logitechFront = webcam(1);                  %Seleccionando webcam usb 1
logitechFront.Resolution = '1280x960';      %Resolución de la webcam en 1280x960
pause(5);
caraTest = snapshot(logitechFront);         %Captura de imagen
delete(logitechFront);                      %Eliminando conexión con webcam
clear logitechFront;                        %Limpiando variable global

logitechUp = webcam(3);                     %Seleccionando webcam usb 3
logitechUp.Resolution = '960x720';          %Resolución de la webcam en 960x720
pause(5);
baseTest = snapshot(logitechUp);            %Captura de imagen
delete(logitechUp);                         %Eliminando conexión con webcam
clear logitechUp;                           %Limpiando variable global

%Recortando imágenes de muestra
caraTest = imcrop(caraTest, recorteCara);
```

```
baseTest = imcrop(baseTest, recorteBase);

%Proceso extracción de features y reconocimiento
%-------------------------------------------------------------------------
%Extracción de los HOG features de cada imagen
trainingFeaturesFace = extractHOGFeatures(caraTest, 'CellSize', cellSize);
trainingFeaturesBase = extractHOGFeatures(baseTest, 'CellSize', cellSize);

%Conversión a tablas de los HOG features
trainingDataFace = array2table(trainingFeaturesFace);
trainingDataBase = array2table(trainingFeaturesBase);

%Reconocimiento de la cara y la cabeza del pistón con los
%clasificadores entrenados
prediccionCara = clasificadorCara.predictFcn(trainingDataFace);
prediccionBase = clasificadorBase.predictFcn(trainingDataBase);

%Envío del diámetro y la altura estándar al Arduino
%-------------------------------------------------------------------------
%Si el reconocimiento concuerda en ambos clasificadores
if prediccionCara == prediccionBase
    prediccionSerie = prediccionCara;

    %Encontrando la posición en la lista de pistones de la serie reconocida
    posicion = find(pistonesToyota==string(prediccionSerie));

    %Diámetro estándar de la serie reconocida
    diametroSTD = diametrosToyota(posicion);

    %Altura estándar de la serie reconocida
    alturaTotal = alturasToyota(posicion);

    %Envío de las medidas estándar a través del puerto
    fwrite(puerto, strcat(diametroSTD, ','));
    pause(1);
    fwrite(puerto, strcat(alturaTotal, ','));
    pause(1);
else
    prediccionSerie = "NA";
end

%Despliegue de las medidas estándar en cajas de texto
%-------------------------------------------------------------------------
%Despliegue de la serie de motor reconocida
set(handles.edit2, 'String', string(prediccionSerie));

%Despliegue de la imagen de muestra de la cara del pistón
axes(handles.axes9);
```

```
imshow(caraTest);

%Despliegue de la imagen de muestra de la cabeza del pistón
axes(handles.axes11);
imshow(baseTest);

if predicciónSerie~="NA"
    %Búsqueda del diseño gráfico de la cara del pistón
    %en la base de datos
    formaCara = imread(strcat('D:\Documentos\Mis Documentos\Trabajo de...
    Graduacion\Interfaz Matlab\Formas Catalogo\Cara\',int2str(posicion),'.jpg'));

    %Búsqueda del diseño gráfico de la cabeza del pistón
    %en la base de datos
    formaBase = imread(strcat('D:\Documentos\Mis Documentos\Trabajo de...
    Graduacion\Interfaz Matlab\Formas Catalogo\Base\',int2str(posicion),'.jpg'));

    %Despliegue de los diseños en los gráficos
    axes(handles.axes10);
    imshow(formaCara);
    axes(handles.axes12);
    imshow(formaBase);

    %Despliegue del diámetro estándar
    set(handles.edit3, 'String', string(diametrosToyota(posicion)));

    %Despliegue de la altura estándar
    set(handles.edit12, 'String', string(alturasToyota(posicion)));

    %Despliegue del ancho estándar del anillo 1
    set(handles.edit5, 'String', string(anillos1(posicion)));

    %Despliegue del ancho estándar del anillo 2
    set(handles.edit6, 'String', string(anillos2(posicion)));

    %Despliegue del ancho estándar del anillo 3
    set(handles.edit7, 'String', string(anillos3(posicion)));

    %Calculando la página donde se encuentra la serie de motor
    %reconocida en el catálogo Teikin
    if (round(posicion/4) - posicion/4) == 0
        set(handles.edit8, 'String', string(posicion/4));
    else
        if (round(posicion/4) - posicion/4) < 0
            set(handles.edit8, 'String', string(round(posicion/4) + 1));
        else
            set(handles.edit8, 'String', string(round(posicion/4)));
        end
```

```
    end
end

%Enviando datos a través del puerto
fwrite(puerto, 'p');

%Activando botones
%--------------------------------------------------------------------------
set(handles.pushbutton5, 'Enable', 'on');     %Botón salir
set(handles.pushbutton13, 'Enable', 'on');    %Botón medir diámetro
set(handles.pushbutton16, 'Enable', 'on');    %Botón nuevo análisis
```

17.3.6. Botón medir diámetro

```
% --- Se ejecuta al presionar el pushbutton13
function pushbutton13_Callback(hObject, eventdata, handles)
%Uso de variables globales para la comunicación con el puerto,
%registro del diámetro medido, medida estándar del diámetro,
%predicción de la serie de  motor
%--------------------------------------------------------------------------
global puerto;
global diametro;
global pistonesToyota;
global  prediccionSerie;
global diametrosToyota;

%Desactivando botones
%--------------------------------------------------------------------------
set(handles.pushbutton5, 'Enable', 'off');    %Botón salir
set(handles.pushbutton13, 'Enable', 'off');   %Botón medir diámetro
set(handles.pushbutton16, 'Enable', 'off');   %Botón nuevo análisis

%Proceso de medición en Arduino
%--------------------------------------------------------------------------
%Enviando datos a través del puerto
fwrite(puerto, 'a');

%Esperando a que termine Arduino
pause(55);

%Enviando datos a través del puerto
fwrite(puerto, 'z');

%Recibiendo la medición del diámetro
diametro =  str2double(fgets(puerto));

%Despliegue del diámetro y cálculo de sobremedida
```

```
%--------------------------------------------------------------------------
%Despliegue diámetro medido
set(handles.edit1, 'String', string(diametro));

%Encontrando la posición en la lista de pistones de la serie reconocida
posicion =  find(pistonesToyota==string(prediccionSerie));

%Diámetro estándar de la serie reconocida
diametroSTD = str2double(diametrosToyota(posicion));

%Diferencia de los diámetros para el cálculo de la sobremedida
sobremedida = abs(diametroSTD - diametro);

%El diámetro de un pistón puede variar de su medida estándar en 4 medidas
%cada medida es conocida como STD, 0.50, 0.75, 1.00
if sobremedida<=0.10
      set(handles.edit4, 'String', "STD");
elseif sobremedida>0.10 && sobremedida<=0.20
  set(handles.edit4, 'String', "0.25"); elseif
    sobremedida>0.20 &&sobremedida<=0.30
      set(handles.edit4, 'String', "0.50");
elseif sobremedida>0.30 &&sobremedida<=0.40
      set(handles.edit4, 'String', "0.75");
elseif sobremedida>0.40
       set(handles.edit4, 'String', "1.00");
end

%Activando botones
%--------------------------------------------------------------------------
set(handles.pushbutton5, 'Enable', 'on');     %Botón salir
set(handles.pushbutton12, 'Enable', 'on');    %Botón medir anillos
set(handles.pushbutton16, 'Enable', 'on');    %Botón nuevo análisis
```

17.3.7. Botón medir anillos

```
% --- Se ejecuta al presionar el pushbutton12
function pushbutton12_Callback(hObject, eventdata, handles)
%Uso de variables globales para la comunicación con el puerto,
%lista de las series de pistones, medida estándar del diámetro,
%predicción de la serie de motor
%--------------------------------------------------------------------------
global puerto;
global  prediccionSerie;
global pistonesToyota;
global diametrosToyota;

%Desactivando botones
```

```
%-------------------------------------------------------------------------
set(handles.pushbutton12, 'Enable', 'off');  %Botón medir anillos
set(handles.pushbutton5, 'Enable', 'off');   %Botón salir
set(handles.pushbutton16, 'Enable', 'off');  %Botón nuevo análisis

%Proceso de posicionamiento y captura de imagen lineal en Arduino
%-------------------------------------------------------------------------
%Enviando datos a través del puerto
fwrite(puerto, 'e');

%Esperando a que termine Arduino
pause(45);

%Recibiendo los pixeles de la cámara en dos partes
cadena = fscanf(puerto)
fwrite(puerto, 'z');
pause(3);
cadena2 = fscanf(puerto)
pause(1);

%Despliegue de pixeles en el gráfico de la cámara TSL1401CL
%-------------------------------------------------------------------------
%Conversión de string a enteros
mapeoCamara = str2num(cadena);
mapeoCamara2 = str2num(cadena2);

%Mapeando valores en un rango de 0 a 1 para el color de la línea
mapeoCamara = mapfun(mapeoCamara, 0, 1023, 0, 1);
mapeoCamara2 = mapfun(mapeoCamara2, 0, 1023, 0, 1);

%Ploteando 128 líneas verticales con un ancho de 20
%de color variable entre 0-1 y grsor de 1
axes(handles.axes7);
for i=4:131
    X = [i i];
    Y = [5  25];
    line(X,Y, 'Color', [0.49  0.18  0.56], 'LineWidth', 1);
end

%Aplicación de un  thresholding según la serie de motor reconocida
%La intensidad de luz reflejada por cada tipo de pistón es variable
%En donde 1 representa un color blanco y 0 un color negro
for i=4:131
    X = [i i];
    Y = [5  25];
    if i<104
        %Selección de un rango de pixeles
        if i<56
```

```
        mapeoCamara(i-3) = 1;
    end

    %Thresholding para la serie 4AFE-II
    if prediccionSerie == "4AFE-II 4AG"
        mapeoCamara(98) = 1;
        mapeoCamara(99) = 1;
        mapeoCamara(100) = 1;
        mapeoCamara(101) = 1;
        mapeoCamara(102) = 1;
        mapeoCamara(103) = 1;
        if mapeoCamara(i-3) > 0.8
            mapeoCamara(i-3) = 1;
        else
            mapeoCamara(i-3) = 0;
        end
    %Thresholding para la serie 3RZ
    elseif prediccionSerie == "3RZ-FE 3RZ-FZ"
        if mapeoCamara(i-3) > 0.25
            mapeoCamara(i-3) = 1;
        else
            mapeoCamara(i-3) = 0;
        end
    %Thresholding para la serie 4AFE-III
    elseif prediccionSerie == "4AFE-III"
        mapeoCamara(98) = 1;
        mapeoCamara(99) = 1;
        mapeoCamara(100) = 1;
        mapeoCamara(101) = 1;
        mapeoCamara(102) = 1;
        mapeoCamara(103) = 1;
        if mapeoCamara(i-3) > 0.6
            mapeoCamara(i-3) = 1;
        else
            mapeoCamara(i-3) = 0;
        end
    %Thresholding para la serie 3E
    elseif prediccionSerie == "3E 3EE"
        if i>89 || i<59
            mapeoCamara(i-3) = 1;
        end
        if mapeoCamara(i-3) > 0.5
            mapeoCamara(i-3) = 1;
        else
            mapeoCamara(i-3) = 0;
        end
    %Thresholding para la serie 1ZZFE
    elseif prediccionSerie == "1ZZFE"
```

```
            if i<65
                mapeoCamara(i-3) = 1;
            end
            ifmapeoCamara(i-3) > 0.7
                mapeoCamara(i-3) = 1;
            else
                mapeoCamara(i-3) = 0;
            end
        % Thresholding para la serie 22R-E
        elseif prediccionSerie == "22R-E"
            if mapeoCamara(i-3) > 0.6
                mapeoCamara(i-3) = 1;
            else
                mapeoCamara(i-3) = 0;
            end
        %Thresholding para la serie 3VZE
        elseif prediccionSerie == "3VZE"
            if mapeoCamara(i-3) > 0.6
                mapeoCamara(i-3) = 1;
            else
                mapeoCamara(i-3) = 0;
            end
        %Thresholding para la serie 21R
        elseif prediccionSerie == "21R 21RC"
            if mapeoCamara(i-3) > 0.7
                mapeoCamara(i-3) = 1;
            else
                mapeoCamara(i-3) = 0;
            end
        %Thresholding para la serie 22R
        elseif prediccionSerie == "22R"
            if mapeoCamara(i-3) > 0.5
                mapeoCamara(i-3) = 1;
            else
                mapeoCamara(i-3) = 0;
            end
        %Thresholding para la serie 20R
        elseif prediccionSerie == "20R"
            if mapeoCamara(i-3) > 0.5
                mapeoCamara(i-3) = 1;
            else
                mapeoCamara(i-3) = 0;
            end
        end
        color = [0+mapeoCamara(i-3) 0+mapeoCamara(i-3) 0+mapeoCamara(i-3)];
    else
        if prediccionSerie == "22R"
            if mapeoCamara2(i-103) > 0.5 || i>115
```

```
                mapeoCamara2(i-103)  = 1;
            else
                mapeoCamara2(i-103)  = 0;
            end
        else
            mapeoCamara2(i-103) = 1;
        end
        color = [0+mapeoCamara2(i-103)  0+mapeoCamara2(i-103)...
                 0+mapeoCamara2(i-103)];
    end
    line(X,Y, 'Color', color, 'LineWidth', 1);
end

%Cálculo del ancho de cada ranura y despliegue en cajas de texto
%-------------------------------------------------------------------------
%Encontrando la posición en la lista de pistones de la serie reconocida
posicion =  find(pistonesToyota==string(prediccionSerie));

%Diámetro estándar de la serie reconocida
diametroSTD = diametrosToyota(posicion);

%Distancia entre la punta del palpador del comparador y el centro
%del pistón dentro del área de trabajo
X = 65;

Pc = 128;                   %Resolución de la cámara TSL1401CL
f = 14;                     %Distancia focal del lente
angulo = 114;               %Ángulo de visión de la cámara

%Distancia entre el lente y la superficie del pistón
xt = X - (str2double(diametroSTD)/2);

%Cálculo del campo de visión
FOV = 2*xt*sin(angulo/2);

%Tamaño por cada pixel en mm
Tpp = FOV/Pc;

%Variable para el almacenamiento de los pixeles negros y blancos
  pixeles = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
             0 0 0 0 0 0 0 0];

%Se recorre las dos listas de pixeles para guardar los datos en una lista
for i=1:128
    if i<101
```

```
        pixeles(i) = mapeoCamara(i);
    else
        pixeles(i) = mapeoCamara2(i-100);
    end
end

pixelesA1 = 0;              %Cantidad de pixeles para el anillo 1
pixelesA2 = 0;              %Cantidad de pixeles para el anillo 2
pixelesA3 = 0;              %Cantidad de pixeles para el anillo 3
pixelAnterior = 1;          %Indica si hay un pixel negro detrás
cuentaA1 = 0;               %Permite el conteo de los pixeles del anillo 1
cuentaA2 = 0;               %Permite el conteo de los pixeles del anillo 2
cuentaA3 = 0;               %Permite el conteo de los pixeles del anillo 3
for i=1:128
    %Si el pixel actual es negro
    if pixeles(i)==0
        %Si el pixel anterior es blanco y no se ha comenzado la cuenta
        %de ninguno de los anillos
        if pixelAnterior==1 && pixelesA1==0 && pixelesA2==0 && pixelesA3==0
            cuentaA1 = 1;
        %Si el pixel anterior es blanco, ya se ha iniciado la cuenta del
        %anillo 1 pero no de los anillos 2 y 3
        elseif pixelAnterior==1 && pixelesA1>0 && pixelesA2==0 && pixelesA3==0
            cuentaA2 = 1;
            cuentaA1 = 0;
        %Si el pixel anterior es blanco, se ha iniciado la cuenta de los
        %anillos 1 y 2 pero no del anillo 3
        elseif pixelAnterior==1 && pixelesA1>0 && pixelesA2>0 && pixelesA3==0
            cuentaA3 = 1;
            cuentaA2 = 0;
        end

        %Cuenta anillo 1
        if cuentaA1 == 1
            pixelesA1 = pixelesA1 + 1;
        %Cuenta anillo 2
        elseif cuentaA2 == 1
            pixelesA2 = pixelesA2 + 1;
        %Cuenta anillo 3
        elseif cuentaA3 == 1
            pixelesA3 = pixelesA3 + 1;
        end
    end

    %El pixel anterior es ahora el pixel actual
    pixelAnterior = pixeles(i);
end
```

```
%Cálculo del ancho de cada anillo según la cantidad de pixeles negros
anchoA1 = pixelesA1 * Tpp;
anchoA2 = pixelesA2 * Tpp;
anchoA3 = pixelesA3 * Tpp;

%Despliegue de las mediciones en cajas de texto
set(handles.edit9, 'String', string(anchoA1));
set(handles.edit10, 'String', string(anchoA2));
set(handles.edit11, 'String', string(anchoA3));

%Proceso de posicionamiento y escaneo de las ranuras en el Arduino
%-------------------------------------------------------------------------
%Enviando datos a través del puerto
fwrite(puerto, 'z');

%Esperando a que termine Arduino
pause(135);

%Recibiendo la nube de puntos del escaneo en  4 partes
cadena = fscanf(puerto);
fwrite(puerto, 'z');
pause(3);
cadena2 = fscanf(puerto);
fwrite(puerto, 'z');
pause(3);
cadena3 = fscanf(puerto);
fwrite(puerto, 'z');
pause(3);
cadena4 = fscanf(puerto);
pause(1);

%Despliegue de la nube de puntos en el gráfico del sensor GY-VL53L0X
%-------------------------------------------------------------------------
%Conversión de string a entero
mapeoLaser = str2num(cadena);
mapeoLaser2 = str2num(cadena2);
mapeoLaser3 = str2num(cadena3);
mapeoLaser4 = str2num(cadena4);

%Mapeando valores en un rango de 5 a 285 para el ancho de la línea
mapeoLaser = mapfun(mapeoLaser, 50, 330, 5, 285);
 mapeoLaser2 =   mapfun(mapeoLaser2, 50, 330, 5, 285);
 mapeoLaser3 =   mapfun(mapeoLaser3, 50, 330, 5, 285);
 mapeoLaser4 =   mapfun(mapeoLaser4, 50, 330, 5, 285);

%Ploteando 350 líneas horizontales con un ancho de 280
%de color blanco y grosor de 1
axes(handles.axes6);
```

```
for i=5:355
    X = [5 285];
    Y = [i i];
    line(X,Y, 'Color', [0.64 0.08 0.18], 'LineWidth', 1);
end

%Ploteando los 350 puntos del escaneo láser, 100 puntos por
%cada lista
for i=1:350
    if i<101
        X = [5 mapeoLaser(i)];
    elseif i>100 && i<201
        X = [5 mapeoLaser2(i-100)];
    elseif i>200 && i<301
        X = [5 mapeoLaser3(i-200)];
    elseif i>300 && i<401
        X = [5 mapeoLaser4(i-300)];
    end
    Y = [i+5 i+5];
    line(X,Y, 'Color', 'white', 'LineWidth', 1);
end

pause(5);
%Enviando datos a través del puerto
fwrite(puerto, 'z');

%Activando botones
%--------------------------------------------------------------------------
set(handles.pushbutton5, 'Enable', 'on');     %Botón salir
set(handles.pushbutton4, 'Enable', 'on');     %Botón identificar desgaste
set(handles.pushbutton16, 'Enable', 'on');    %Botón nuevo análisis
```

17.3.8. Botón identiftcar desgaste

```
% --- Se ejecuta al presionar el pushbutton4
function pushbutton4_Callback(hObject, eventdata, handles)
%Uso de variables globales para la comunicación con el puerto, uso de la
%cámra del microscopio digital
%--------------------------------------------------------------------------
global microscopio;
global puerto;

microscopio = webcam(4);              %Seleccionando cámara usb 4
recorteDesgaste = [110 35 810 610]; %Área de recorte de la imagen

%Desactivando botones
%--------------------------------------------------------------------------
```

```
set(handles.pushbutton4, 'Enable', 'off');   %Botón identificar desgaste
set(handles.pushbutton5, 'Enable', 'off');   %Botón salir
set(handles.pushbutton16, 'Enable', 'off');  %Botón nuevo análisis

%Proceso de giro del pistón mientras se toman las imágenes de las 8 regiones
%-------------------------------------------------------------------------

%PRIMERA REGIÓN
%Enviando datos a través del puerto
fwrite(puerto, 'b');

%Esperando a que el pistón gire a la primera región
pause(35);

%Captura de la imagen de la primera región
faldaPiston = snapshot(microscopio);

%Identificación del desgaste con la función
desgaste = analisisDesgaste(faldaPiston);

%Recorte de la imagen
desgaste = imcrop(desgaste, recorteDesgaste);

%Se guarda la imagen capturada y la del procesamiento de la función
imwrite(faldaPiston, 'D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\analisisDesgaste\faldaPiston1.jpg');
imwrite(desgaste, 'D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\analisisDesgaste\desgaste1.jpg');

%SEGUNDA REGIÓN
%Enviando datos a través del puerto
fwrite(puerto, 'z');

%Esperando a que el pistón gire a la segunda región
pause(15);

%Captura de la imagen de la segunda región
faldaPiston = snapshot(microscopio);

%Identificación del desgaste con la función
desgaste = analisisDesgaste(faldaPiston);

%Recorte de la imagen
desgaste = imcrop(desgaste, recorteDesgaste);

%Se guarda la imagen capturada y la del procesamiento de la función
imwrite(faldaPiston, 'D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\analisisDesgaste\faldaPiston2.jpg');
```

```
imwrite(desgaste, 'D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\analisisDesgaste\desgaste2.jpg');

%TERCERA REGIÓN
%Enviando datos a través del puerto
fwrite(puerto, 'z');

%Esperando a que el pistón gire a la tercera región
pause(8);

%Captura de la imagen de la tercera región
faldaPiston = snapshot(microscopio);

%Identificación del desgaste con la función
desgaste = analisisDesgaste(faldaPiston);

%Recorte de la imagen
desgaste = imcrop(desgaste, recorteDesgaste);

%Se guarda la imagen capturada y la del procesamiento de la función
imwrite(faldaPiston, 'D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\analisisDesgaste\faldaPiston3.jpg');
imwrite(desgaste, 'D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\analisisDesgaste\desgaste3.jpg');

%CUARTA REGIÓN
%Enviando datos a través del puerto
fwrite(puerto, 'z');

%Esperando a que el pistón gire a la cuarta región
pause(8);

%Captura de la imagen de la cuarta región
faldaPiston = snapshot(microscopio);

%Identificación del desgaste con la función
desgaste = analisisDesgaste(faldaPiston);

%Recorte de la imagen
desgaste = imcrop(desgaste, recorteDesgaste);

%Se guarda la imagen capturada y la del procesamiento de la función
imwrite(faldaPiston, 'D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\analisisDesgaste\faldaPiston4.jpg');
imwrite(desgaste, 'D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\analisisDesgaste\desgaste4.jpg');

%QUINTA REGIÓN
```

```
%Enviando datos a través del puerto
fwrite(puerto, 'z');

%Esperando a que el pistón gire a la quinta región
pause(10);

%Captura de la imagen de la quinta región
faldaPiston = snapshot(microscopio);

%Identificación del desgaste con la función
desgaste = analisisDesgaste(faldaPiston);

%Recorte de la imagen
desgaste = imcrop(desgaste, recorteDesgaste);

%Se guarda la imagen capturada y la del procesamiento de la función
imwrite(faldaPiston, 'D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\analisisDesgaste\faldaPiston5.jpg');
imwrite(desgaste, 'D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\analisisDesgaste\desgaste5.jpg');

%SEXTA REGIÓN
%Enviando datos a través del puerto
fwrite(puerto, 'z');

%Esperando a que el pistón gire a la sexta región
pause(8);

%Captura de la imagen de la sexta región
faldaPiston = snapshot(microscopio);

%Identificación del desgaste con la función
desgaste = analisisDesgaste(faldaPiston);

%Recorte de la imagen
desgaste = imcrop(desgaste, recorteDesgaste);

%Se guarda la imagen capturada y la del procesamiento de la función
imwrite(faldaPiston, 'D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\analisisDesgaste\faldaPiston6.jpg');
imwrite(desgaste, 'D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\analisisDesgaste\desgaste6.jpg');

%SÉPTIMA REGIÓN
%Enviando datos a través del puerto
fwrite(puerto, 'z');

%Esperando a que el pistón gire a la séptima región
```

```
pause(8);

%Captura de la imagen de la séptima región
faldaPiston = snapshot(microscopio);

%Identificación del desgaste con la función
desgaste = analisisDesgaste(faldaPiston);

%Recorte de la imagen
desgaste = imcrop(desgaste, recorteDesgaste);

%Se guarda la imagen capturada y la del procesamiento de la función
imwrite(faldaPiston, 'D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\analisisDesgaste\faldaPiston7.jpg');
imwrite(desgaste, 'D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\analisisDesgaste\desgaste7.jpg');

%Posicionamiento en home del pistón y el microscopio en Arduino
%--------------------------------------------------------------------------
%Enviando datos a través del puerto
fwrite(puerto, 'z');

%Esperando a que Arduino termine
pause(35);

%Despliegue de las imágenes en los gráficos
%--------------------------------------------------------------------------
%Despliegue de la primera región
axes(handles.axes14);
imshow('D:\Documentos\Mis Documentos\Trabajo de Graduacion\Interfaz...
Matlab\analisisDesgaste\desgaste1.jpg');

%Despliegue de la segunda región
axes(handles.axes15);
imshow('D:\Documentos\Mis Documentos\Trabajo de Graduacion\Interfaz...
Matlab\analisisDesgaste\desgaste2.jpg');

%Despliegue de la tercera región
axes(handles.axes16);
imshow('D:\Documentos\Mis Documentos\Trabajo de Graduacion\Interfaz...
Matlab\analisisDesgaste\desgaste3.jpg');

%Despliegue de la cuarta región
axes(handles.axes17);
imshow('D:\Documentos\Mis Documentos\Trabajo de Graduacion\Interfaz...
Matlab\analisisDesgaste\desgaste4.jpg');

%Despliegue de la quinta región
```

```
axes(handles.axes18);
imshow('D:\Documentos\Mis Documentos\Trabajo de Graduacion\Interfaz...
Matlab\analisisDesgaste\desgaste5.jpg');

%Despliegue de la sexta región
axes(handles.axes19);
imshow('D:\Documentos\Mis Documentos\Trabajo de Graduacion\Interfaz...
Matlab\analisisDesgaste\desgaste6.jpg');

%Despliegue de la séptima región
axes(handles.axes20);
imshow('D:\Documentos\Mis Documentos\Trabajo de Graduacion\Interfaz...
Matlab\analisisDesgaste\desgaste7.jpg');

delete(microscopio);            %Eliminando conexión con el microscopio
clear microscopio;              %Limpiando variable global

fwrite(puerto, 'z');            %Enviando datos a través del puerto
pause(1);

%Activando botones
%--------------------------------------------------------------------------
set(handles.pushbutton5, 'Enable', 'on');     %Botón salir
set(handles.pushbutton16, 'Enable', 'on');    %Botón nuevo análisis
```

Botón agregar serie

```
%--- Se ejecuta al presionar el pushbutton7
function pushbutton7_Callback(hObject, eventdata, handles)
%Uso de variables globales para la comunicación con el puerto, lista de las
%series de pistones
%--------------------------------------------------------------------------
global pistonesToyota;
global puerto;

%Desactivando botones
%--------------------------------------------------------------------------
set(handles.pushbutton3, 'Enable', 'off');    %Botón reconocer serie
set(handles.pushbutton5, 'Enable', 'off');    %Botón salir
set(handles.pushbutton7, 'Enable', 'off');    %Botón agregar serie
set(handles.pushbutton19, 'Enable', 'off');   %Botón colocar pistón
set(handles.pushbutton16, 'Enable', 'off');   %Botón nuevo análisis

%Actualizando lista de series por registrar de la base de datos
%--------------------------------------------------------------------------
%Lectura del nombre de las carpetas de cada serie de la base de datos
for i=1:length(directorio)
```

```
    cadena = strcat(cadena,',',directorio(i).name);
end

%Converisón a celdas
labels = strsplit(cadena,',');

%Eliminación de celdas basura '' '.' '..'
labels(1:3) = [];

listPistonesToyota = "";          %Lista actualizada de series por registrar
serieAnterior = "";               %Indica el nombre de la serie anterior

%Se recorre la lista de las series de pistones
for i=1:length(pistonesToyota)
    %Se recorre la lista del nombre de las carpetas de la base de datos
    for j=1:length(labels)

        %Si el nombre de la serie de la lista no se encuentra en las
        %carpetas de la base de datos y no se repite el nombre en
        %la lista actualizada que se está creando
        comparacionSeries  = strcmp(pistonesToyota(i),labels(j));
        comparacionRepetido = serieAnterior ~= pistonesToyota(i);
        if not(comparacionSeries) && (comparacionRepetido)
            %Creación de la lista actualizada con los nombres de las series
            %de pistones que faltan por registrar en la base de datos
            listPistonesToyota = listPistonesToyota + newline + pistonesToyota(i);
        end
        %La serie actual pasa a ser la anterior en el recorrido
        serieAnterior = pistonesToyota(i);
    end
end

%Despliegue de la lista actualizada con los nombres de las series
%------------------------------------------------------------------------
set(handles.listbox1, 'Enable', 'on');   %Activar la caja de lista
set(handles.listbox1, 'String', "");     %Limpiar la caja de lista
set(handles.listbox1, 'String', listPistonesToyota);      %Despliege de nombres

%Enviando datos a través del puerto
fwrite(puerto, 'c');
```

17.3.9. Botón salir

```
% --- Se ejecuta al presionar el pushbutton5
function pushbutton5_Callback(hObject, eventdata, handles)
%Finalización de la comunicación con el puerto y rutina por defecto de la
%función CloseRequestFcn que deshabilita el control HandleVisibility que
```

```
%hace visible la ventana
%--------------------------------------------------------------------------
global puerto;
fclose(puerto);
delete(puerto);
set(0,'ShowHiddenHandles','on')
delete(get(0,'Children'))

%Limpiando todas las variables utilizadas
clear all;

%Cierre de la ventana
close;
```

17.3.10. Caja de lista registro de series

```
% --- Se ejecuta al cambiar la selección en listbox1
function listbox1_Callback(hObject, eventdata, handles)
%Uso de la variable global que almacena el nombre de la serie seleccionado
%--------------------------------------------------------------------------
global serieLabel;

%Proceso de selección de serie a registrar
%--------------------------------------------------------------------------
%Obteniendo la lista de las series de la listbox
contenido = get(hObject, 'String');

%Obteniendo la posición de la serie que se ha seleccionado
index = get(hObject, 'Value');

%Obteniendo el nombre de la serie seleccionada con la posición
serieLabel = string(contenido(index));

%Despliegue del nombre de la serie seleccionada en una etiqueta
serieLabel = newline + serieLabel;
set(handles.text10, 'String', serieLabel);

%Se almacena el nombre de la serie a registrar
serieLabel = string(contenido(index));

%Activando boton
%--------------------------------------------------------------------------
set(handles.pushbutton8, 'Enable', 'on');     %Botón sesión de fotos
```

17.3.11. Botón sesión de fotos

```
% --- Se ejecuta al presionar el pushbutton8
function pushbutton8_Callback(hObject, eventdata, handles)
%Uso de la variable global que almacena el nombre de la serie seleccionado
%--------------------------------------------------------------------------
global serieLabel;

%Creación de la nueva carpeta en la base de datos que contendrá las imágenes
%de las 12 muestras de pistones de la serie seleccionada
%--------------------------------------------------------------------------
%Creación de la carpeta para la cabeza del pistón
mkdir('D:\Documentos\Mis Documentos\Trabajo de Graduacion\Interfaz...
Matlab\dataBase',char(serieLabel));

%Creación de la carpeta para la cara del pistón
mkdir('D:\Documentos\Mis Documentos\Trabajo de Graduacion\Interfaz...
Matlab\dataCara',char(serieLabel));


%Desactivando y activando botones
%--------------------------------------------------------------------------
set(handles.pushbutton8, 'Enable', 'off');   %Botón sesión de fotos
set(handles.pushbutton9, 'Enable', 'on');    %Botón tomar fotos
set(handles.pushbutton11, 'Enable', 'on');   %Botón cancelar
set(handles.listbox1, 'Enable', 'off');      %Caja de lista de series
```

17.3.12. Botón tomar fotos

```
% --- Se ejecuta al presionar el pushbutton9
function pushbutton9_Callback(hObject, eventdata, handles)
%Uso de variables globales para la comunicación con el puerto, uso de las
%cámaras logitech, nombre de la serie a registrar, cantidad de fotos tomadas,
%área de recorte de las imágenes
%--------------------------------------------------------------------------
global cantFotos;
global logitechFront;
global logitechUp;
global serieLabel;
global puerto;
global recorteCara;
global recorteBase;

%Desactivando botones
%--------------------------------------------------------------------------
set(handles.pushbutton9, 'Enable', 'off');   %Botón tomar fotos
```

```
set(handles.pushbutton11, 'Enable', 'off');  %Botón cancelar

%Colocación del pistón en la base del mecanismo
%--------------------------------------------------------------------------
fwrite(puerto, 'p');          %Activando el mecanismo para sujetar el pistón
pause(8);                     %Retraso para lograr colocar el pistón
fwrite(puerto, 'c');          %Indicando seguimiento del proceso
fwrite(puerto, 'v');          %Girando el pistón sobre su eje
pause(10);                    %Esperando al Arduino para empezar la primera
                              %tanda de fotos

%Proceso de captura de imágenes para la primera tanda
%--------------------------------------------------------------------------
logitechFront = webcam(1);                %Seleccionando webcam usb 1
logitechFront.Resolution = '1280x960';    %Resolución de la webcam en 1280x960
imageCara = snapshot(logitechFront);      %Captura de la imagen
clear('logitechFront');                   %Eliminando conexión con webcam

logitechUp = webcam(3);                   %Seleccionando webcam usb 3
logitechUp.Resolution = '960x720';        %Resolución de la webcam en 960x720
imageBase = snapshot(logitechUp);         %Captura de la imagen
clear('logitechUp');                      %Eliminando conexión con webcam

%Recortando imágenes
imageCaracrop = imcrop(imageCara, recorteCara);
imageBasecrop = imcrop(imageBase, recorteBase);

cantFotos = cantFotos+ 2;                 %Incremento del contador de fotos

%Guardando las imágenes en sus respectivas carpetas para la cara y la cabeza
fileNameC = strcat('D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\dataCara\',serieLabel,'\',int2str(cantFotos/2),'.jpg');
fileNameB = strcat('D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\dataBase\',serieLabel,'\',int2str(cantFotos/2),'.jpg');
imwrite(imageCaracrop, char(fileNameC));
imwrite(imageBasecrop, char(fileNameB));

fwrite(puerto, 'd');                      %Indicando seguimiento del proceso

%Proceso de captura de imágenes para la segunda tanda
%--------------------------------------------------------------------------
pause(10);                    %Esperando al Arduino para empezar la segunda
                              %tanda de fotos

logitechFront = webcam(1);                %Seleccionando webcam usb 1
logitechFront.Resolution = '1280x960';    %Resolución de la webcam en 1280x960
imageCara = snapshot(logitechFront);      %Captura de la imagen
clear('logitechFront');                   %Eliminando conexión con webcam
```

```
logitechUp = webcam(3);                    %Seleccionando webcam usb 3
logitechUp.Resolution = '960x720';         %Resolución de la webcam en 960x720
imageBase = snapshot(logitechUp);          %Captura de la imagen
clear('logitechUp');                       %Eliminando conexión con webcam

%Recortando imágenes
imageCaracrop = imcrop(imageCara, recorteCara);
imageBasecrop = imcrop(imageBase, recorteBase);

cantFotos = cantFotos + 2;                 %Incremento del contador de fotos

%Guardando las imágenes en sus respectivas carpetas para la cara y la cabeza
fileNameC = strcat('D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\dataCara\',serieLabel,'\',int2str(cantFotos/2),'.jpg');
fileNameB = strcat('D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\dataBase\',serieLabel,'\',int2str(cantFotos/2),'.jpg');
imwrite(imageCaracrop, char(fileNameC));
imwrite(imageBasecrop, char(fileNameB));

fwrite(puerto, 'd');                       %Indicando seguimiento del proceso

%Despliegue del contador de fotos en una caja de texto
set(handles.text11, 'String', string(cantFotos));

%Activando botones
%--------------------------------------------------------------------------
set(handles.pushbutton9, 'Enable', 'on');     %Botón tomar fotos
set(handles.pushbutton11, 'Enable', 'on');    %Botón cancelar

%Si ya se tienen un total de 48 fotos de las 12 muestras en la base de datos
%entonces se desactiva la función para seguir tomando fotos
if cantFotos==48
    set(handles.pushbutton9, 'Enable', 'off');
    set(handles.pushbutton10, 'Enable', 'on');
end

fwrite(puerto, 'c');                       %Indicando seguimiento del proceso
```

17.3.13. Botón entrenar clasiftcador

```
% --- Se ejecuta al presionar el pushbutton10
function pushbutton10_Callback(hObject, eventdata, handles)
%Uso de variables globales para el entrenamiento de clasificadores,
%el contador de fotos, número de fotos por pistón, número total de fotos
%en la base de datos, tamaño de celda para los features, comunicación
%con el puerto
```

```
%-------------------------------------------------------------------------
global cantFotos;
global numImages;
global numImagesPP;
global puerto;
global  cellSize;
global clasificadorCara;
global clasificadorBase;

%Desactivando botones
%-------------------------------------------------------------------------
set(handles.pushbutton10, 'Enable', 'off');  %Botón entrenar clasificador
set(handles.pushbutton11, 'Enable', 'off');  %Botón cancelar
set(handles.listbox1, 'Enable', 'off');      %Caja de lista de las series

%Proceso de entrenamiento de los clasificadores
%-------------------------------------------------------------------------
%Definiendo el nombre de las clases con el de las series en la base de datos
myDataFace = imageDatastore('dataCara', 'IncludeSubfolders', true, ...
                            'LabelSource', 'foldernames');
myDataBase = imageDatastore('dataBase', 'IncludeSubfolders', true, ...
                            'LabelSource', 'foldernames');
[myDataTrainFace, ] = myDataFace.splitEachLabel(numImagesPP);
[myDataTrainBase, ] = myDataBase.splitEachLabel(numImagesPP);

%Encontrando las HOG features de una imagen aleatoria
ii = randi(length(myDataFace.Files));
[featureVectorFace, ] = extractHOGFeatures(myDataFace.readimage(ii), ...
                                           'CellSize', [32 32]);
[featureVectorBase, ] = extractHOGFeatures(myDataBase.readimage(ii), ...
                                           'CellSize', [32 32]);

%Contador del total de imágenes en la base de datos
numImages = 192+(cantFotos/2);

%Calculando el la cantidad de features
hogFeatureSizeFace = length(featureVectorFace);
hogFeatureSizeBase = length(featureVectorBase);

%Creación de las listas que contendrán las features de la cabeza y la cara
trainingFeaturesFace = zeros(numImages, hogFeatureSizeFace, 'single');
trainingFeaturesBase = zeros(numImages, hogFeatureSizeBase,  'single');

%Recorriendo todas las imágenes y extrayendo las HOG features para
%crear los datos de entrenamiento
for ii= 1:numImages
    trainingFeaturesFace(ii, :) = extractHOGFeatures(...
    myDataTrainFace.readimage(ii), 'CellSize', cellSize);
```

```
    trainingFeaturesBase(ii, :) = extractHOGFeatures(...
    myDataTrainBase.readimage(ii), 'CellSize', cellSize);
end

%Conversión de las listas a tablas
trainingDataFace = array2table(trainingFeaturesFace);
trainingDataBase   =   array2table(trainingFeaturesBase);

%Definiendo el nombre de cada feature
trainingDataFace.objectType = myDataTrainFace.Labels;
trainingDataBase.objectType  = myDataTrainBase.Labels;

%Llamando a la función que entrena los clasificadores
[clasificadorCara, ] = trainClassifierFace(trainingDataFace);
[clasificadorBase, ] = trainClassifierBase(trainingDataBase);

%Despliegue de la cantidad de series de pistones en la base de datos
set(handles.text13, 'String', "Pistones en BD = "+string(numImages/numImagesPP));

%Reiniciando el contador de fotos
cantFotos = 0;
set(handles.text11, 'String', "0");

%Activando botones
%--------------------------------------------------------------------
set(handles.pushbutton3, 'Enable', 'on');    %Botón reconocer serie
set(handles.pushbutton5, 'Enable', 'on');    %Botón salir
set(handles.pushbutton7, 'Enable', 'on');    %Botón Agregar serie
set(handles.pushbutton19, 'Enable', 'on');   %Botón colocar pistón
set(handles.pushbutton16, 'Enable', 'on');   %Botón nuevo análisis
```

17.3.14. Botón cancelar

```
% --- Se ejecuta al presionar el pushbutton11
function pushbutton11_Callback(hObject, eventdata, handles)
%Uso de variables globales para la comunicación con el puerto, contador de
%fotos tomadas, nombre de la serie a registrar
%--------------------------------------------------------------------
global serieLabel;
global cantFotos;
global puerto;

cantFotos = 0;              %Reinicio del contador de fotos

%Desactivando botones
%--------------------------------------------------------------------
set(handles.pushbutton8, 'Enable', 'off');
```

```
set(handles.pushbutton9, 'Enable', 'off');
set(handles.pushbutton10, 'Enable', 'off');
set(handles.pushbutton11, 'Enable', 'off');

%Limpiando cajas de texto y eliminación de carpetas
%--------------------------------------------------------------------------
%Desactivando la caja de lista de las series
set(handles.listbox1, 'Enable', 'off');

%Limpiando la etiqueta con la serie seleccionada anteriormente
set(handles.text10, 'String', "");

%Despliegue del contador en 0
set(handles.text11, 'String', "0");

%Eliminación de las carpetas con las fotos tomadas de la cabeza y la cara
%del pistón
cmd_rmdir(char(strcat('D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\dataBase\',char(serieLabel))));
cmd_rmdir(char(strcat('D:\Documentos\Mis Documentos\Trabajo de Graduacion...
\Interfaz Matlab\dataCara\',char(serieLabel))));


%Activando botones
%--------------------------------------------------------------------------
set(handles.pushbutton3, 'Enable', 'on');
set(handles.pushbutton5, 'Enable', 'on');
set(handles.pushbutton7, 'Enable', 'on');
set(handles.pushbutton19, 'Enable', 'on');
set(handles.pushbutton16, 'Enable', 'on');

fwrite(puerto, 'z');              %Indicando seguimiento del proceso
```

17.3.15. Botón nuevo análisis

```
% --- Se ejecuta al presionar el pushbutton16
function pushbutton16_Callback(hObject, eventdata, handles)
%Activando botones
%--------------------------------------------------------------------------
set(handles.pushbutton5, 'Enable', 'on');     %Botón salir
set(handles.pushbutton7, 'Enable', 'on');     %Botón Agregar serie
set(handles.pushbutton19, 'Enable', 'on');    %Botón colocar pistón

%Desactivando botones
%--------------------------------------------------------------------------
set(handles.pushbutton3, 'Enable', 'off');    %Botón reconocer serie
set(handles.pushbutton4, 'Enable', 'off');    %Botón identificar desgaste
```

```
set(handles.pushbutton13, 'Enable', 'off');  %Botón medir diámetro
set(handles.pushbutton12, 'Enable', 'off');  %Botón medir anillos

%Limpiando cajas de texto
%-------------------------------------------------------------------------
set(handles.edit1, 'String', "");    %Diámetro del pistón
set(handles.edit2, 'String', "");    %Serie de motor
set(handles.edit3, 'String', "");    %Medida estándar
set(handles.edit4, 'String', "");    %Sobremedida
set(handles.edit5, 'String', "");    %Anillo 1
set(handles.edit6, 'String', "");    %Anillo 2
set(handles.edit7, 'String', "");    %Anillo 3
set(handles.edit8, 'String', "");    %Página en catálogo
set(handles.edit9, 'String', "");    %Anillo 1 cámara
set(handles.edit10, 'String', "");   %Anillo 2 cámara
set(handles.edit11, 'String', "");   %Anillo 3 cámara
set(handles.edit12, 'String', "");   %Altura total

%Cargando imágenes de inicio en los gráficos
%-------------------------------------------------------------------------
axes(handles.axes9);          %Gráfico imagen de muestra cara del pistón
imshow('pistonLogo.png');
axes(handles.axes10);         %Gráfico imagen forma geométrica cara del pistón
imshow('pistonLogo.png');
axes(handles.axes11);         %Gráfico imagen de muestra cabeza del pistón
imshow('pistonLogo.png');
axes(handles.axes12);         %Gráfico imagen forma geométrica cabeza del pistón
imshow('pistonLogo.png');
axes(handles.axes14);         %Gráfico imagen primera region falda del pistón
imshow('pistonLogo.png');
axes(handles.axes15);         %Gráfico imagen segunda region falda del pistón
imshow('pistonLogo.png');
axes(handles.axes16);         %Gráfico imagen tercera region falda del pistón
imshow('pistonLogo.png');
axes(handles.axes17);         %Gráfico imagen cuarta region falda del pistón
imshow('pistonLogo.png');
axes(handles.axes18);         %Gráfico imagen quinta region falda del pistón
imshow('pistonLogo.png');
axes(handles.axes19);         %Gráfico imagen sexta region falda del pistón
imshow('pistonLogo.png');
axes(handles.axes20);         %Gráfico imagen séptima region falda del pistón
imshow('pistonLogo.png');

%Limpiando gráfico del sensor y la cámara
%-------------------------------------------------------------------------
%Gráfico del sensor GY-VL53L0X
axes(handles.axes6);
for i=5:355
```

```
    X = [5 285];
    Y = [i i];
    line(X,Y, 'Color', 'white', 'LineWidth', 1);
end

%Gráfico de la cámara TSL1401CL
axes(handles.axes7);
for i=4:131
    X = [i i];
    Y = [5 25];
    line(X,Y, 'Color', 'white', 'LineWidth', 1);
end
```

17.3.16. Función para la identiftcación de rayaduras

```
function [desgaste] = analisisDesgaste(imagen)

%Conversión a escala de grises
gray = rgb2gray(imagen);

%Aplicando un thresholding adaptativo de 0.45
T = adaptthresh(gray,0.45);

%Binarización de la imagen en escala de grises
BW = imbinarize(gray, T);

%Eliminación de áreas pequeñas
BWA = bwareaopen(BW, 30);

%Relleno de áreas
BWAC = imclose(BWA, strel('disk',2));

%Selección de áreas en el rango de 100 a 7000 pixeles
BWA2 = bwareafilt(BWAC, [100 7000]);

%Desactivando la ventana que muestra la imagen
fig = figure('visible', 'off');
imshow(imagen);
hold on

%Colocando los bordes encima de la imagen original
visboundaries(BWA2, 'color', 'yellow');

%Registrando el directorio de la imagen temporal a guardar
dir = 'D:\Documentos\Mis Documentos\Trabajo de Graduacion\Interfaz Matlab';
dir2 = '\analisisDesgaste\deteccionDesgaste.jpg';
archivo = strcat(dir, dir2);
```

```
%Guardando la imagen temporal en el directorio
saveas(fig, archivo);

%Devolviendo la imagen analizada al programa principal
desgaste   =  imread(archivo);

%Eliminando  imagen temporal
delete(archivo);
end
```

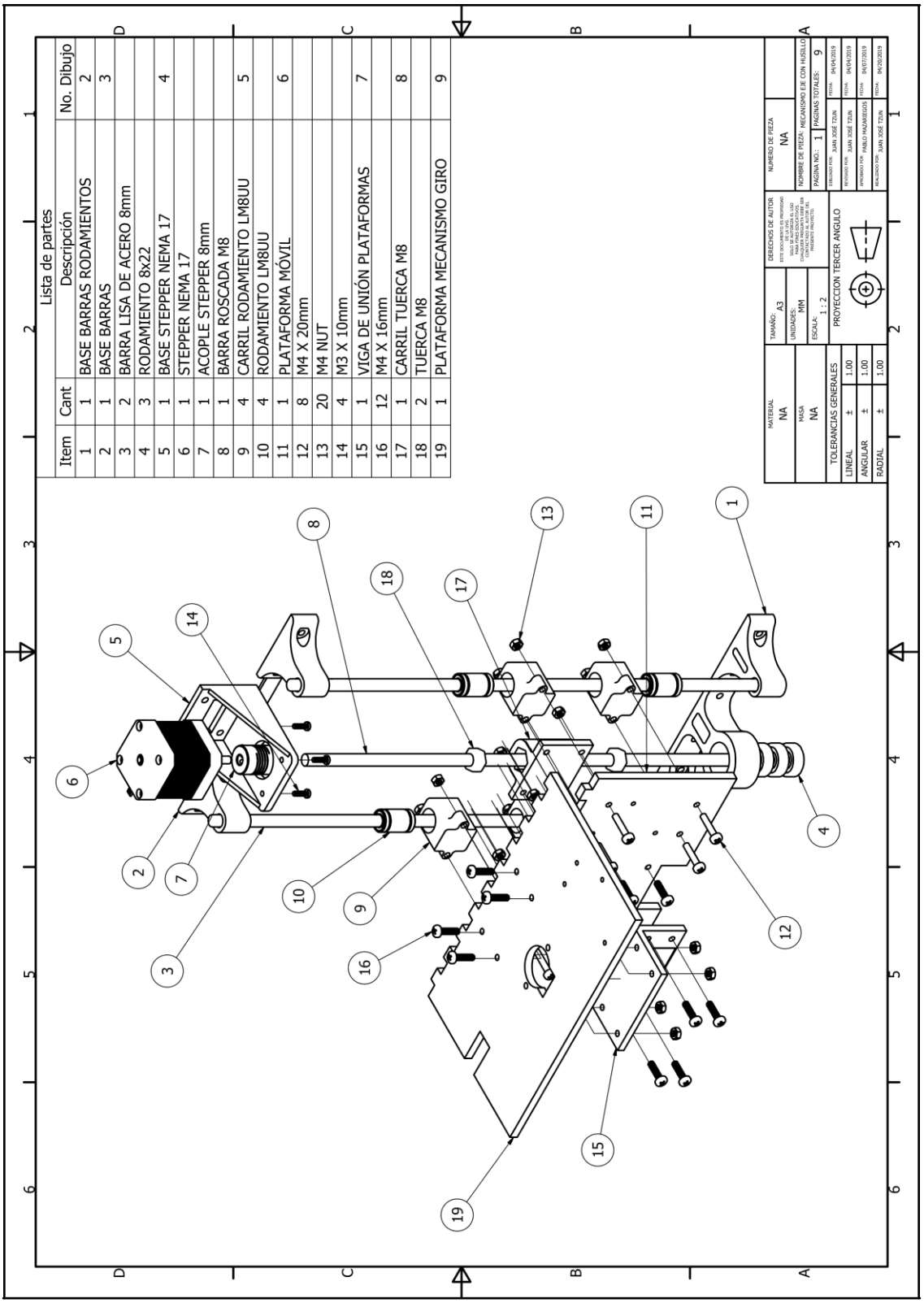
17.3.17. Botón atrás catálogo Teikin

```
% --- Se ejecuta al presionar el pushbutton17
function pushbutton17_Callback(hObject, eventdata, handles)
%Uso  de variables globales lista de las series de pistones, índice de pistón
%que se está visualizando en el  gráfico
%--------------------------------------------------------------------------
global visualizador;
global pistonesToyota;

%Si el contador es mayor que 1 se seguirá decrementando en 1
if visualizador>1
    visualizador = visualizador - 1;
end

%Visualización del recorte de las medidas estándar de la serie de pistón
%seleccionado con el índice
%--------------------------------------------------------------------------
axes(handles.axes13);

%Se extrae de la base de datos el recorte de la serie correspondiente
fotoCatalogo = imread(strcat('D:\Documentos\Mis Documentos\Trabajo de ...
Graduacion\Interfaz Matlab\Catalogo Teikin\',int2str(visualizador),'.jpg'));

%Buscando en la lista de las series el nombre de la serie seleccionada
%según el índice
serie = pistonesToyota(visualizador);

%Despliegue del nombre de la serie
set(handles.text24, 'String', string(serie));

%Despliegue del recorte de la imagen
imshow(fotoCatalogo);
```

17.3.18. Botón adelante catálogo Teikin

```
% --- Se ejecuta al presionar el pushbutton18
function pushbutton18_Callback(hObject, eventdata, handles)
%Uso  de variables globales lista de las series de pistones, índice de pistón
%que se está visualizando en el  gráfico
%--------------------------------------------------------------------------
global visualizador;
global pistonesToyota;

%Si el contador es menor que 140 se seguirá incrementando en 1
if visualizador<140
    visualizador = visualizador + 1;
end

%Visualización del recorte de las medidas estándar de la serie de pistón
%seleccionado con el índice
%--------------------------------------------------------------------------
axes(handles.axes13);

%Se extrae de la base de datos el recorte de la serie correspondiente
fotoCatalogo = imread(strcat('D:\Documentos\Mis Documentos\Trabajo de ...
Graduacion\Interfaz Matlab\Catalogo Teikin\',int2str(visualizador),'.jpg'));

%Buscando en la lista de las series el nombre de la serie seleccionada
%según el índice
serie = pistonesToyota(visualizador);

%Despliegue del nombre de la serie
set(handles.text24, 'String', string(serie));

%Despliegue del recorte de la imagen
imshow(fotoCatalogo);
```

17.4. Planos mecanismo eje con husillo

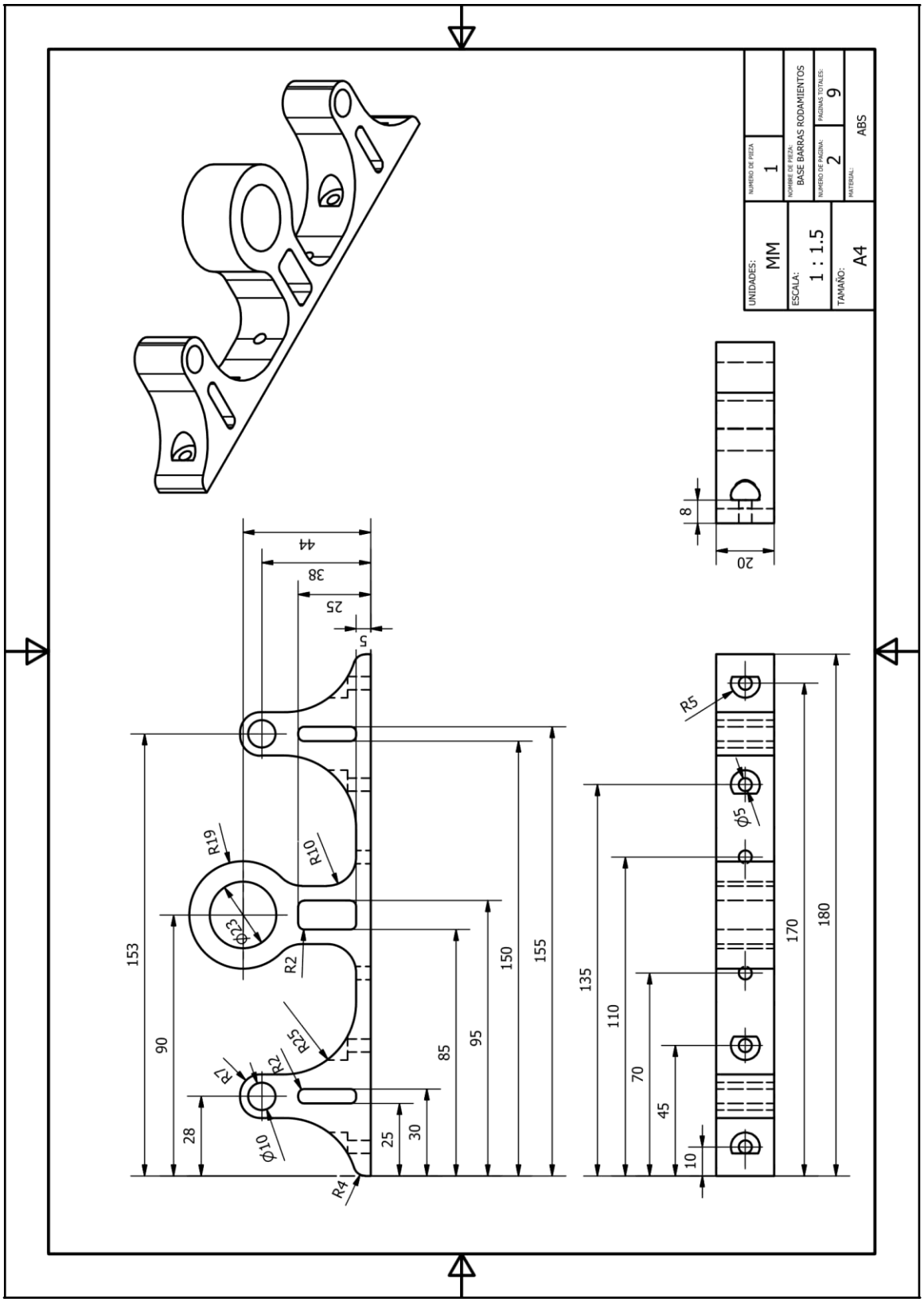


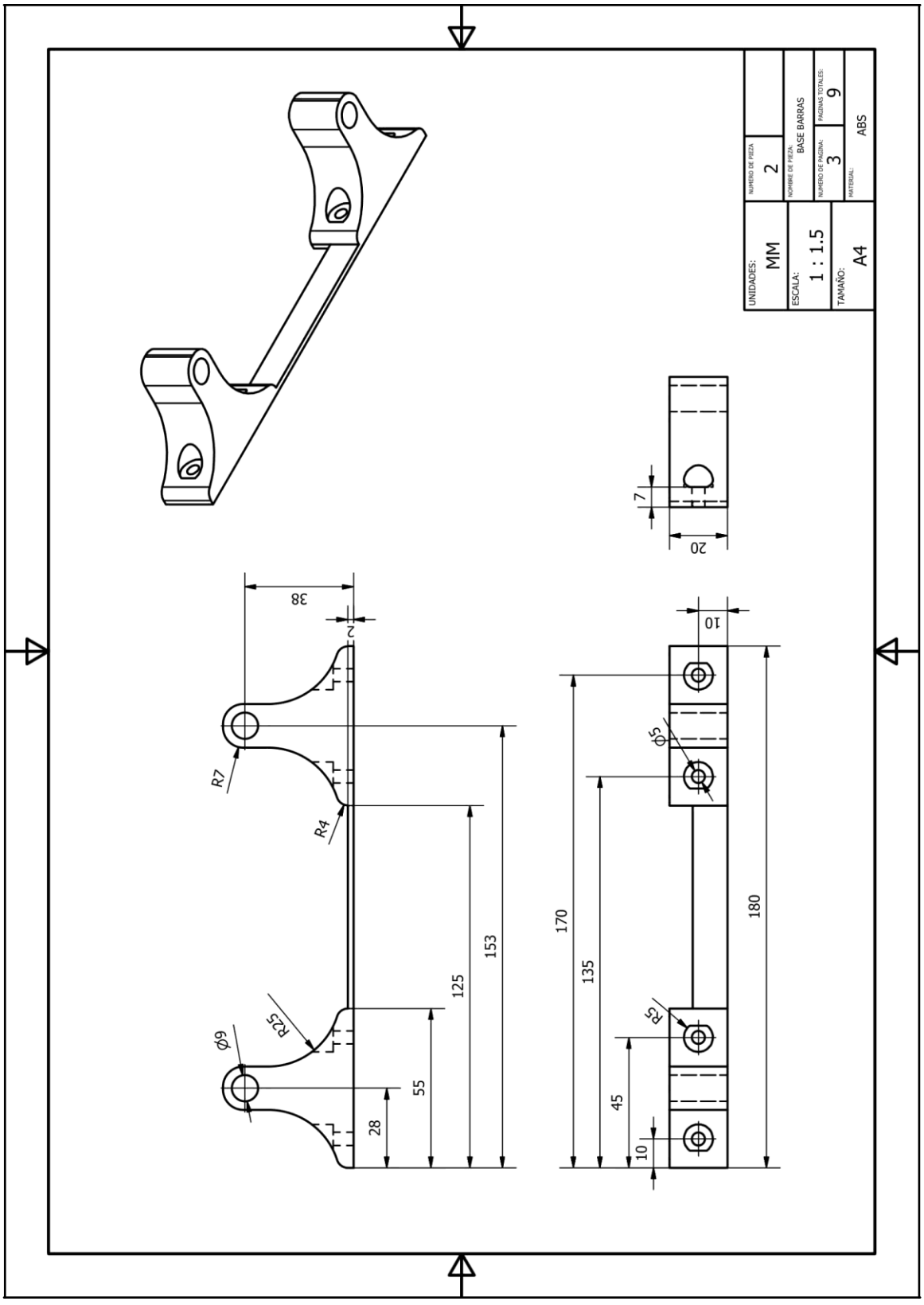
Lista de partes

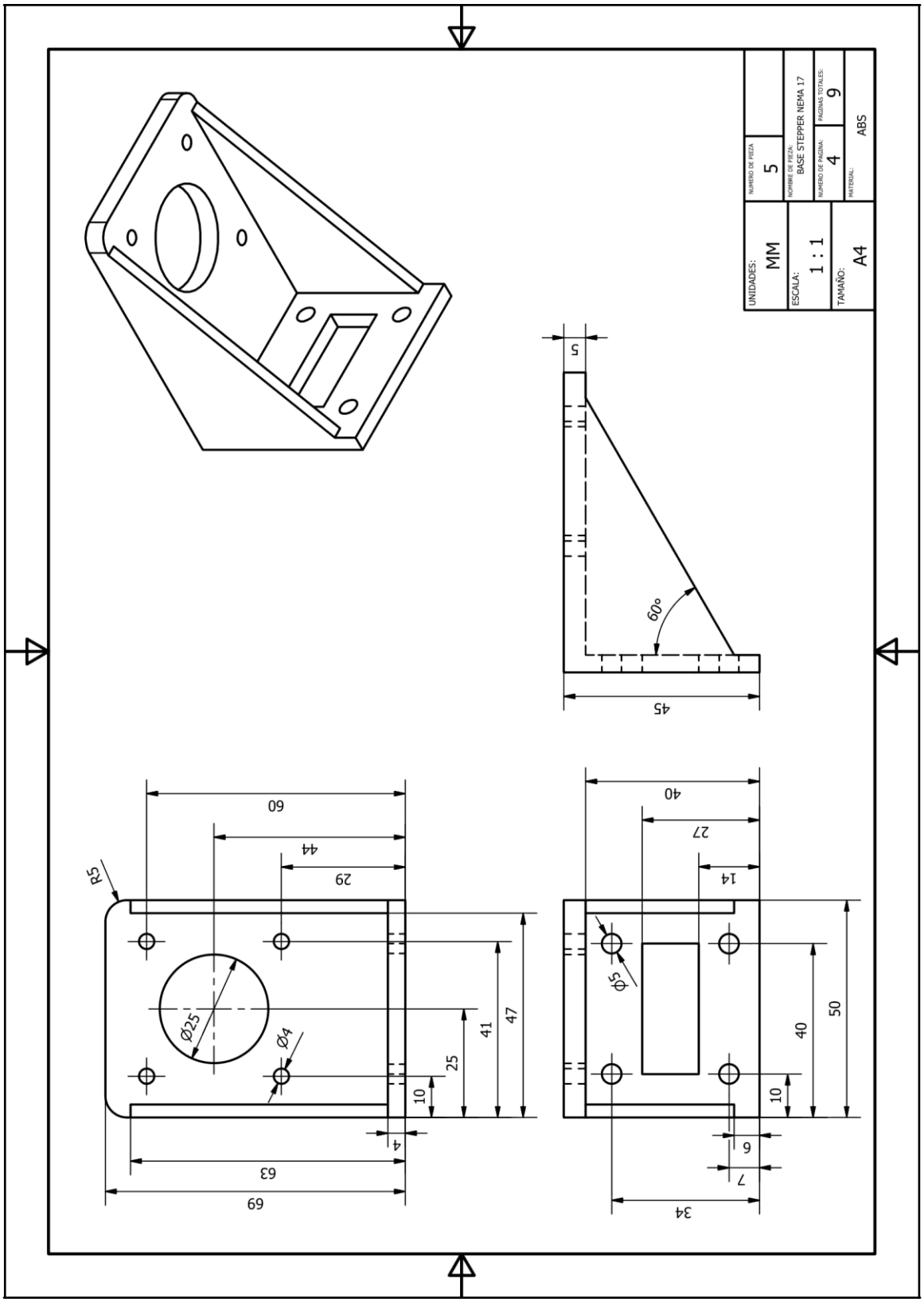
Item	Cant	Descripción	No. Dibujo
1	1	BASE BARRAS RODAMIENTOS	2
2	1	BASE BARRAS	3
3	2	BARRA LISA DE ACERO 8mm	
4	3	RODAMIENTO 8x22	
5	1	BASE STEPPER NEMA 17	4
6	1	STEPPER NEMA 17	
7	1	ACOPLE STEPPER 8mm	
8	1	BARRA ROSCADA M8	
9	4	CARRIL RODAMIENTO LM8UU	5
10	4	RODAMIENTO LM8UU	
11	1	PLATAFORMA MÓVIL	6
12	8	M4 X 20mm	
13	20	M4 NUT	
14	4	M3 X 10mm	
15	1	VIGA DE UNIÓN PLATAFORMAS	7
16	12	M4 X 16mm	
17	1	CARRIL TUERCA M8	8
18	2	TUERCA M8	
19	1	PLATAFORMA MECANISMO GIRO	9

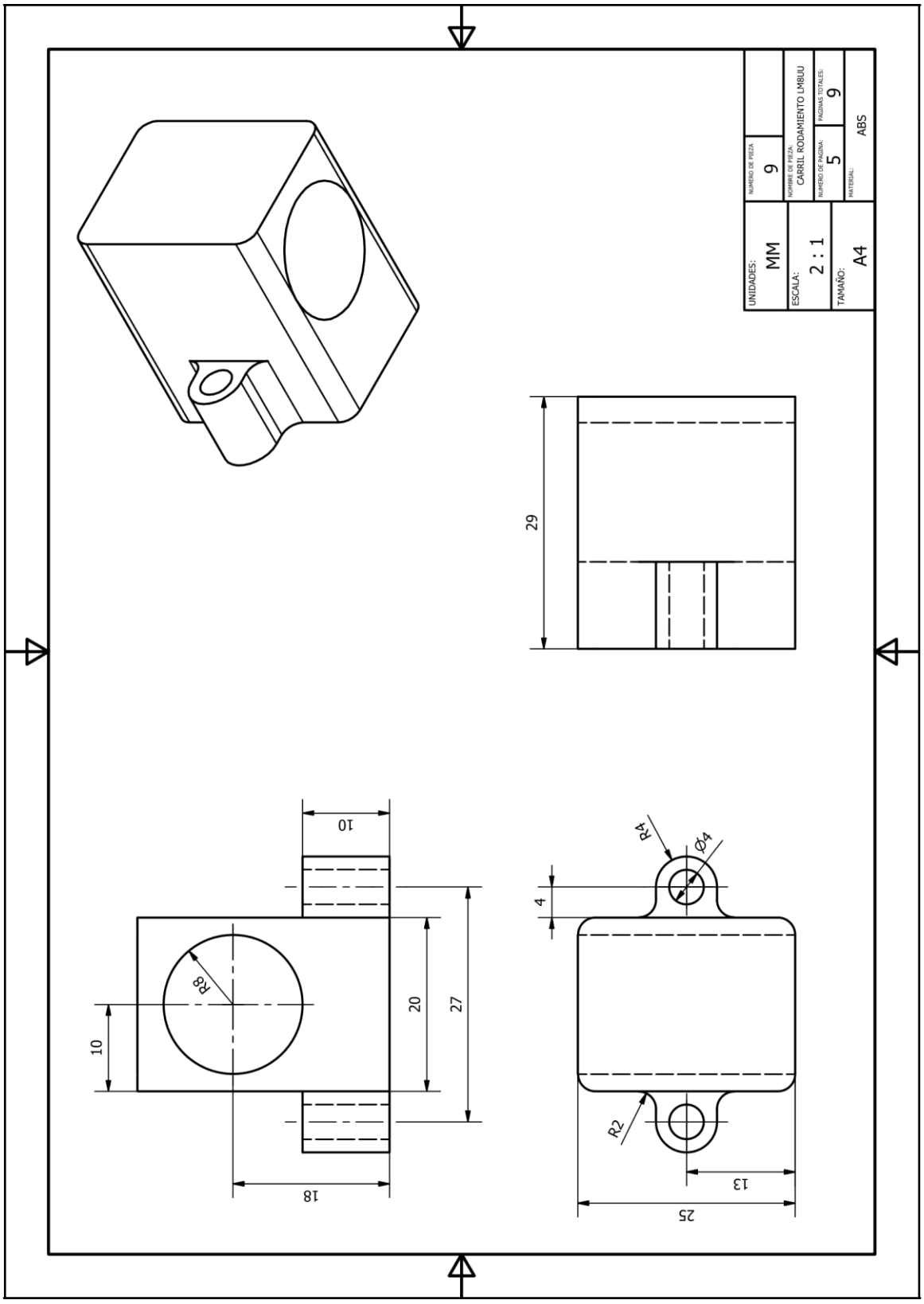
MATERIAL: NA
MASA: NA
TAMAÑO: A3
UNIDADES: MM
ESCALA: 1 : 2
DERECHOS DE AUTOR
NÚMERO DE PIEZA: NA
NOMBRE DE PIEZA: MECANISMO EJE CON HUSILLO
PÁGINA NO.: 1
PÁGINAS TOTALES: 9
PROYECCION TERCER ANGULO
DIBUJADO POR: JUAN JOSÉ TZUN
FECHA: 04/04/2019
REVISADO POR: JUAN JOSÉ TZUN
FECHA: 04/04/2019
APROBADO POR: PABLO PAZARIEGOS
FECHA: 04/07/2019
REALIZADO POR: JUAN JOSÉ TZUN
FECHA: 04/20/2019

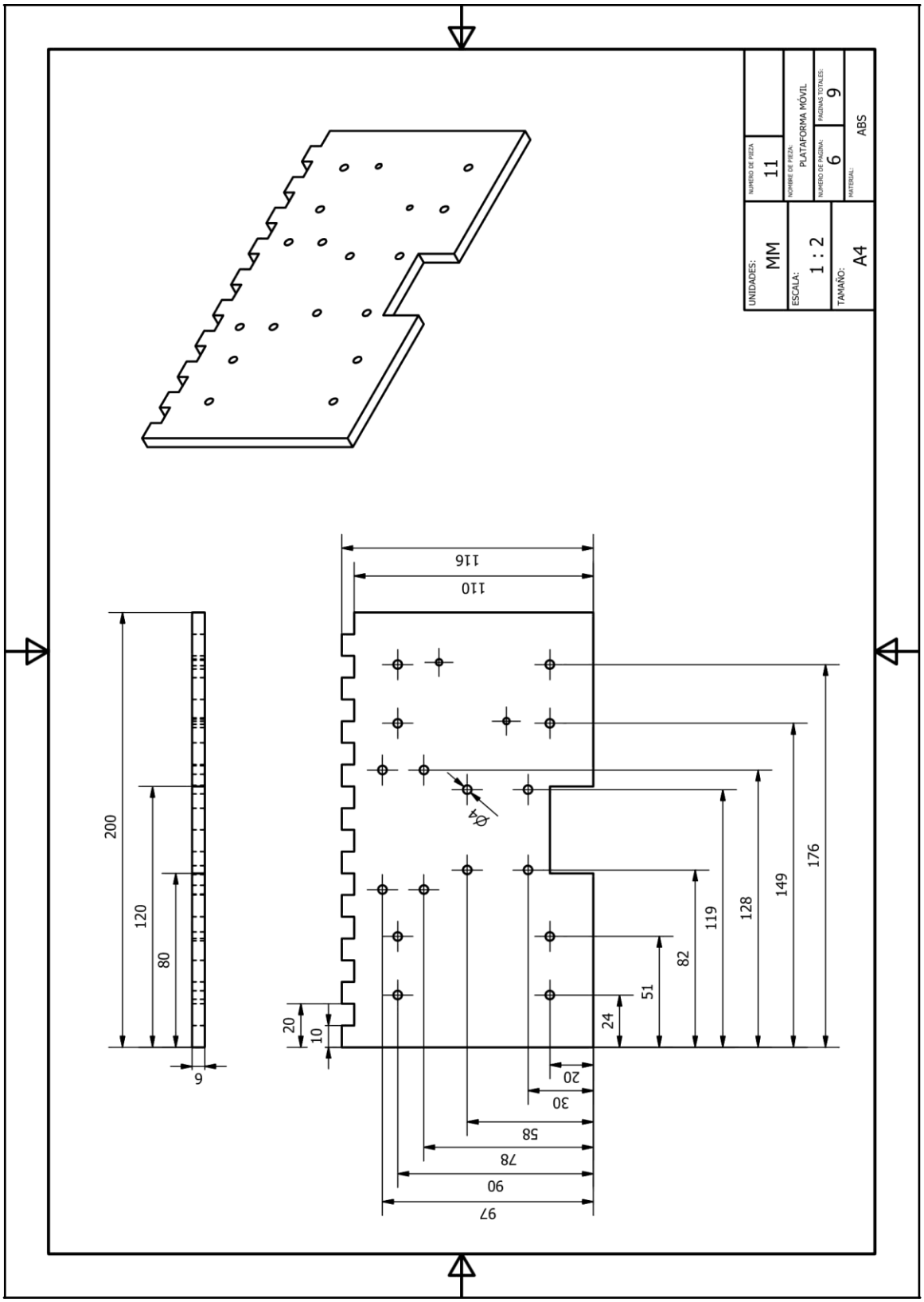
TOLERANCIAS GENERALES
LINEAL ± 1.00
ANGULAR ± 1.00
RADIAL ± 1.00

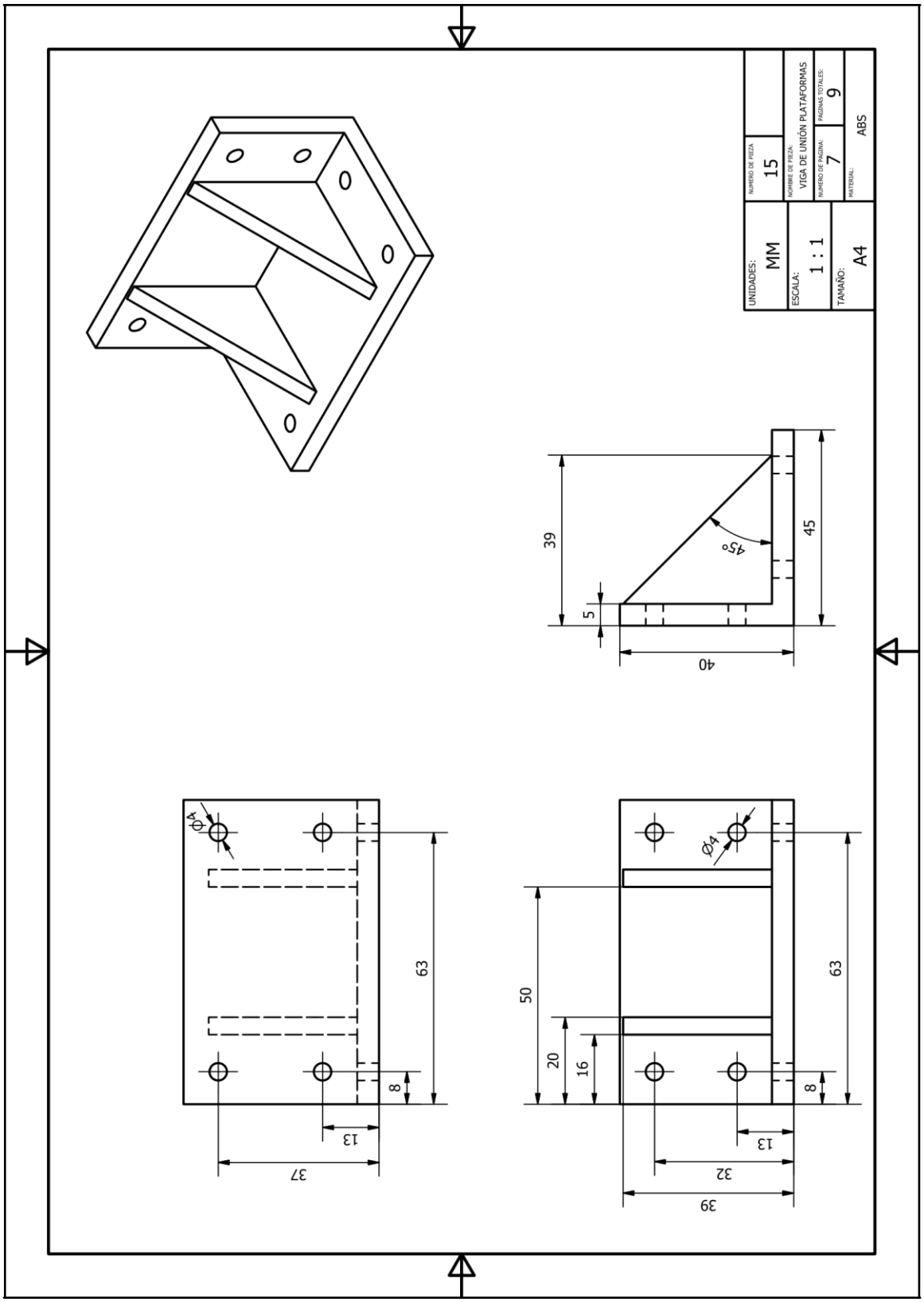

UNIDADES: MM
NUMERO DE PIEZA: 1
NOMBRE DE PIEZA: BASE BARRAS RODAMIENTOS
NUMERO DE PAGINA: 2
PAGINAS TOTALES: 9
ESCALA: 1 : 1.5
TAMAÑO: A4
MATERIAL: ABS
44
38
25
5
153
90
28
R19
R10
ϕ23
R2
R25
R7
ϕ10
R4
85
95
150
155
25
30
8
20
R5
ϕ5
135
110
70
45
10
170
180

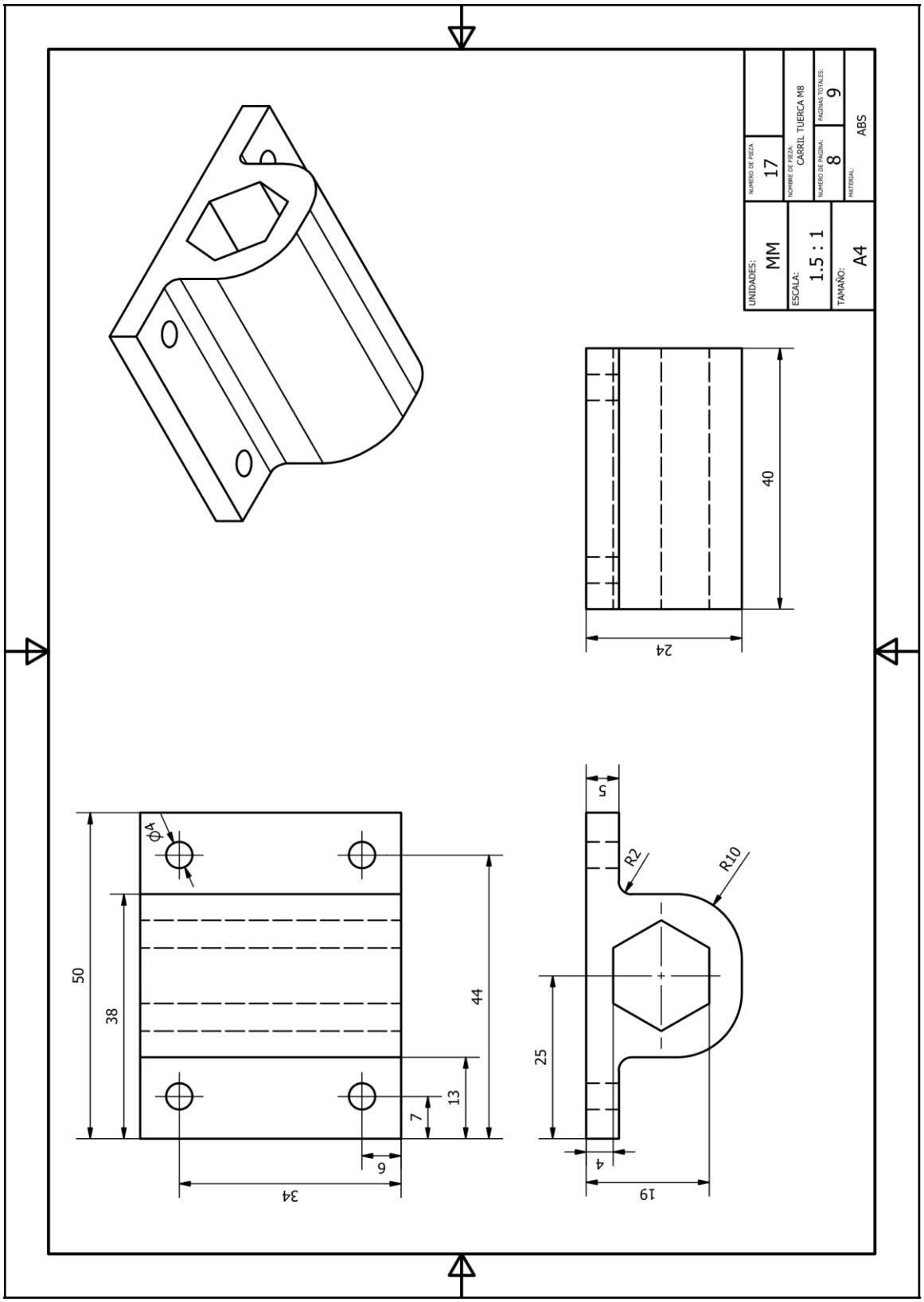

UNIDADES: MM
ESCALA: 1 : 1.5
TAMAÑO: A4
NUMERO DE PIEZA: 2
NOMBRE DE PIEZA: BASE BARRAS
NUMERO DE PAGINA: 3
PAGINAS TOTALES: 9
MATERIAL: ABS
38
2
R7
R4
R25
ϕ9
28
55
125
153
7
20
10
ϕ5
R2
10
45
135
170
180

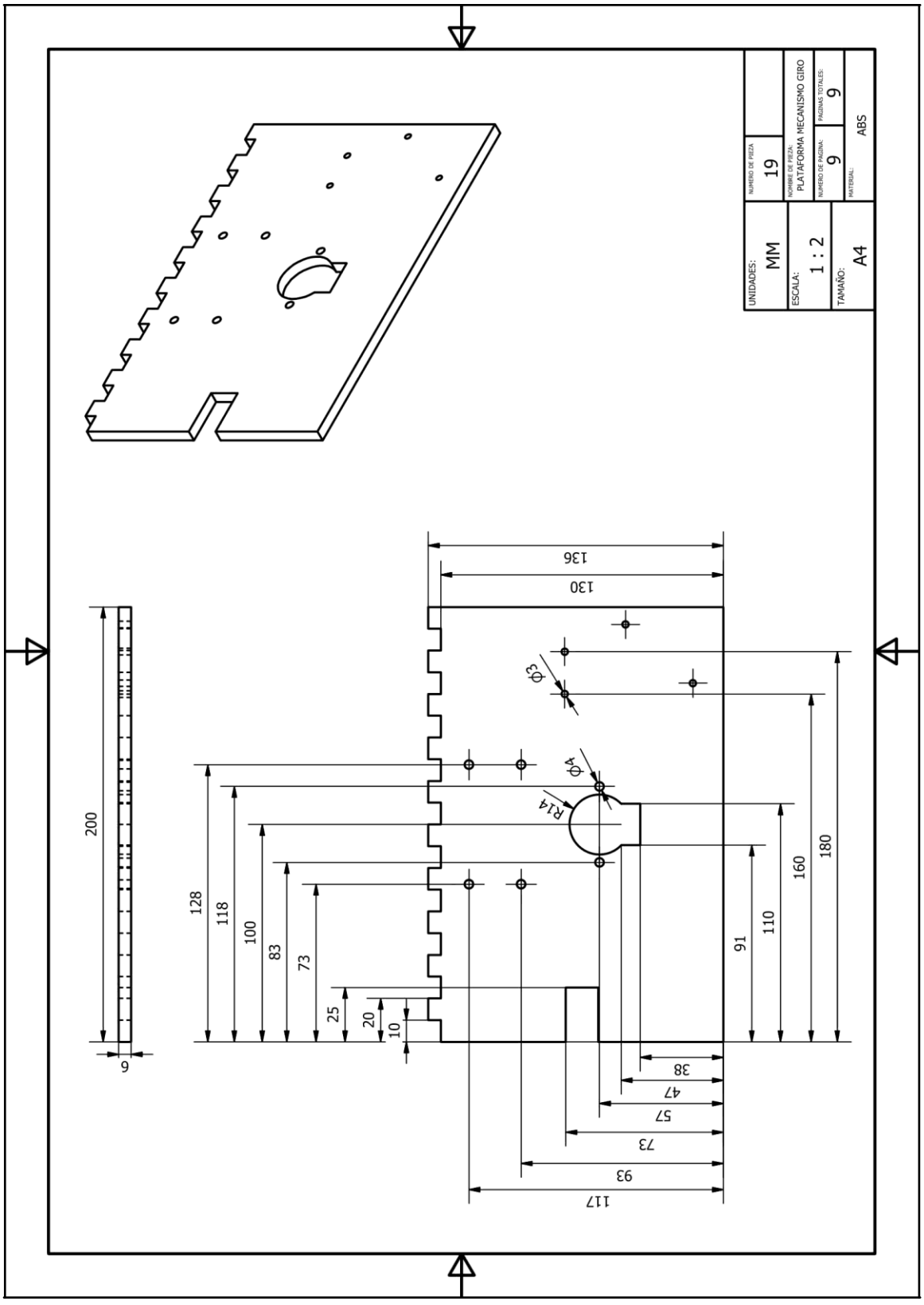

UNIDADES: MM
ESCALA: 1 : 1
TAMAÑO: A4
NUMERO DE PIEZA: 5
NOMBRE DE PIEZA: BASE STEPPER NEMA 17
NUMERO DE PAGINA: 4
PAGINAS TOTALES: 9
MATERIAL: ABS
5
60°
45
60
44
29
R5
Ø25
Ø4
47
41
25
10
4
63
69
40
27
14
Ø5
50
40
10
6
7
34


NUMERO DE PIEZA: 9
NOMBRE DE PIEZA: CARRIL RODAMIENTO LMBUU
NUMERO DE PAGINA: 5
PAGINAS TOTALES: 9
MATERIAL: ABS
UNIDADES: MM
ESCALA: 2 : 1
TAMAÑO: A4
29
10
R8
10
20
27
18
R4
Ø4
4
R2
13
25

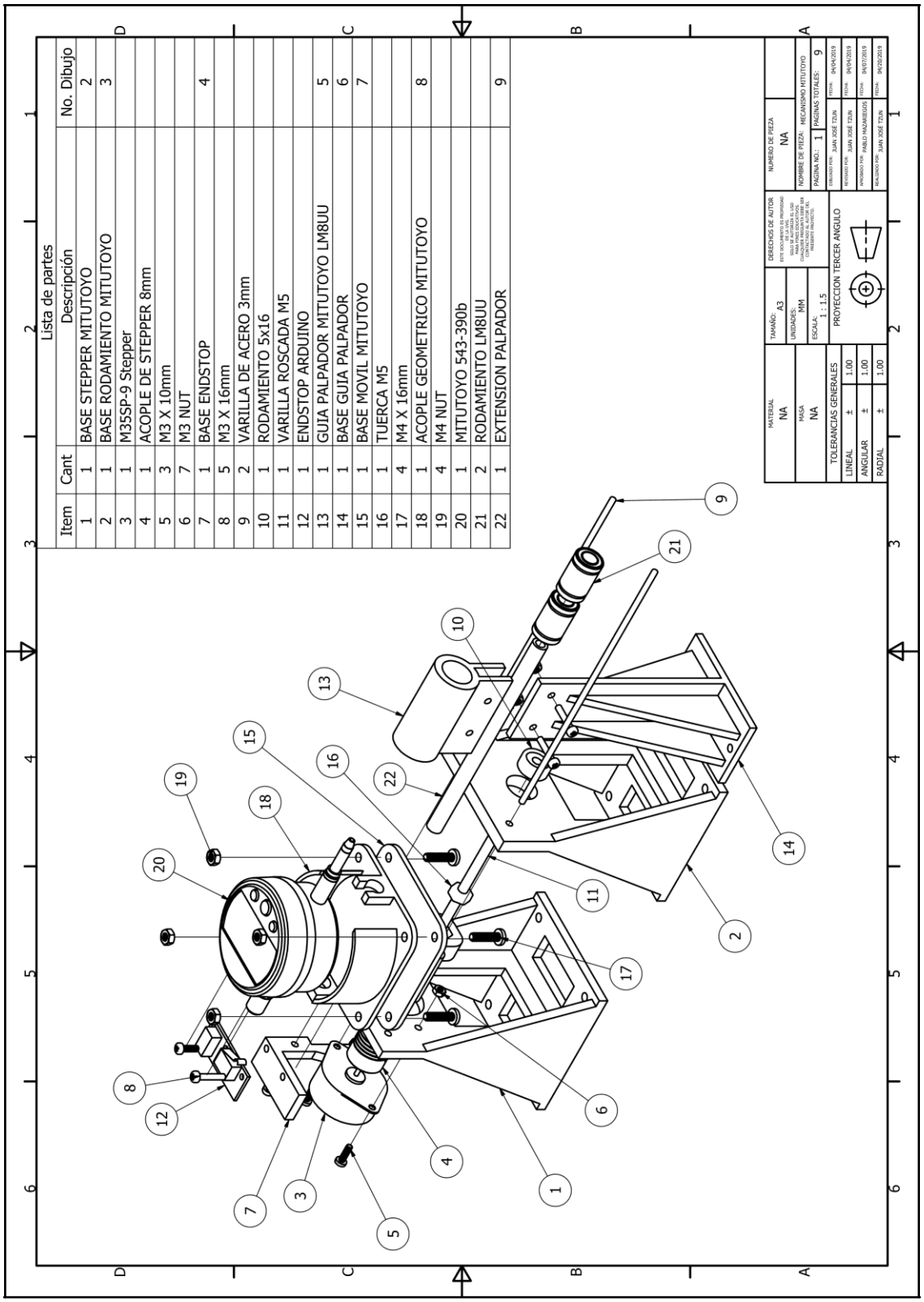

UNIDADES: MM
ESCALA: 1 : 2
TAMAÑO: A4
NUMERO DE PIEZA: 11
NOMBRE DE PIEZA: PLATAFORMA MÓVIL
NUMERO DE PAGINA: 6
PAGINAS TOTALES: 9
MATERIAL: ABS
116
110
200
120
80
9
Ø4
176
149
128
119
82
51
24
20
10
20
30
58
78
90
97

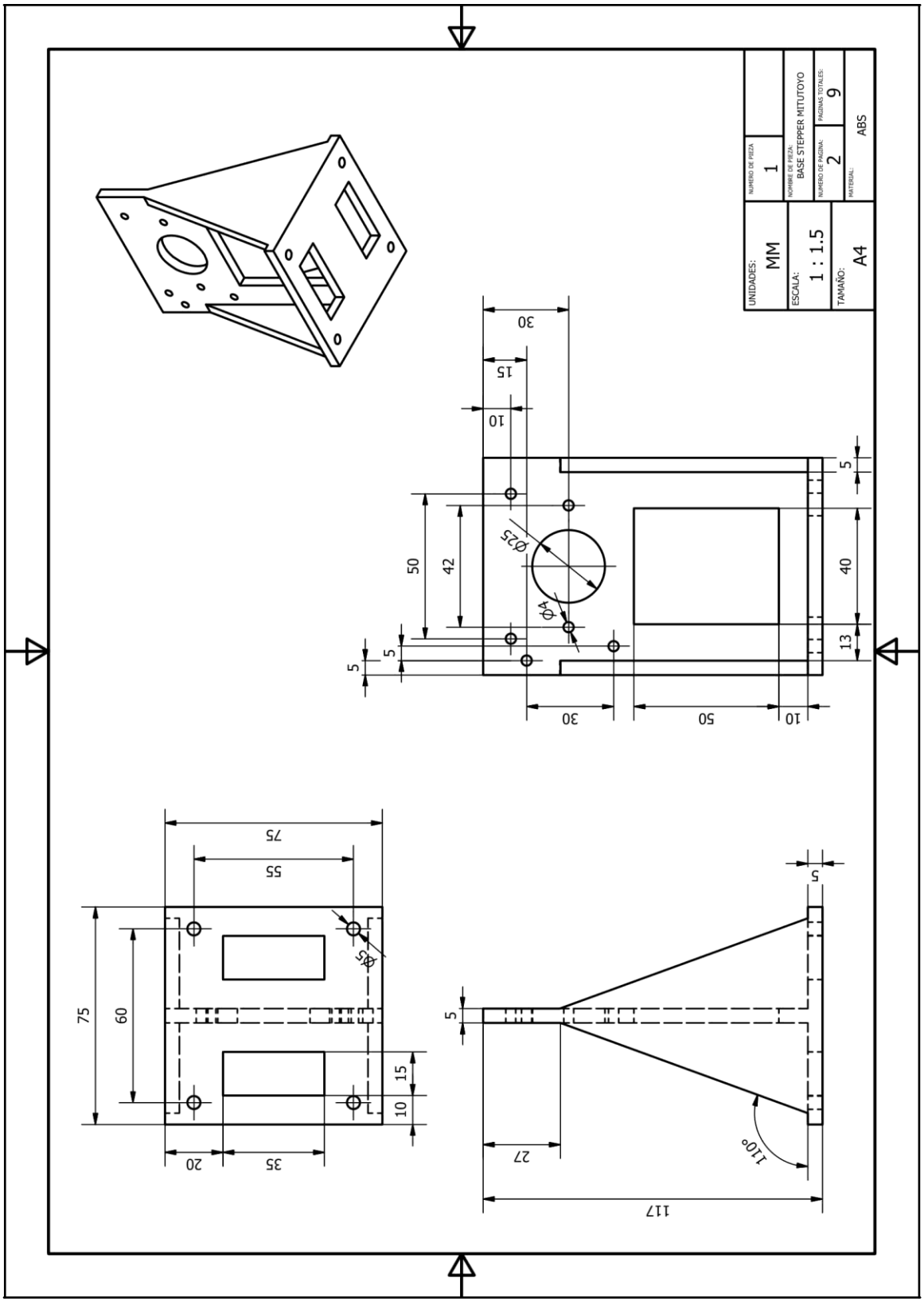

NÚMERO DE PIEZA:
15
NOMBRE DE PIEZA:
VIGA DE UNIÓN PLATAFORMAS
NÚMERO DE PÁGINA:
7
PÁGINAS TOTALES:
9
MATERIAL:
ABS
UNIDADES:
MM
ESCALA:
1 : 1
TAMAÑO:
A4
39
45°
45
5
40
ϕ4
63
8
13
37
Ø4
50
20
16
63
8
13
32
39

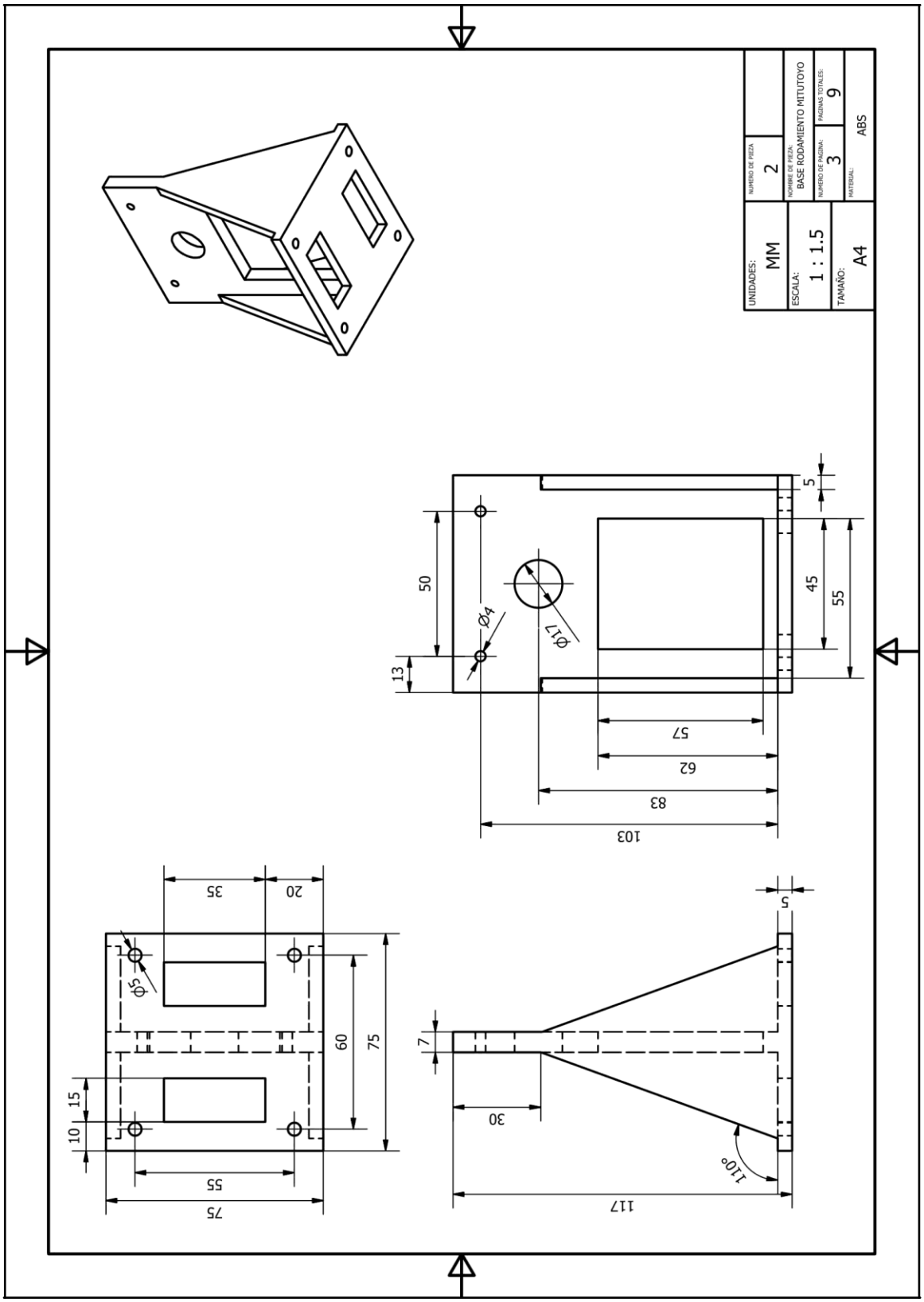

UNIDADES: MM
ESCALA: 1.5 : 1
TAMAÑO: A4
NUMERO DE PIEZA: 17
NOMBRE DE PIEZA: CARRIL TUERCA M8
NUMERO DE PÁGINA: 8
PÁGINAS TOTALES: 9
MATERIAL: ABS
40
24
ϕA
50
38
44
13
7
6
34
5
R2
R10
25
4
19

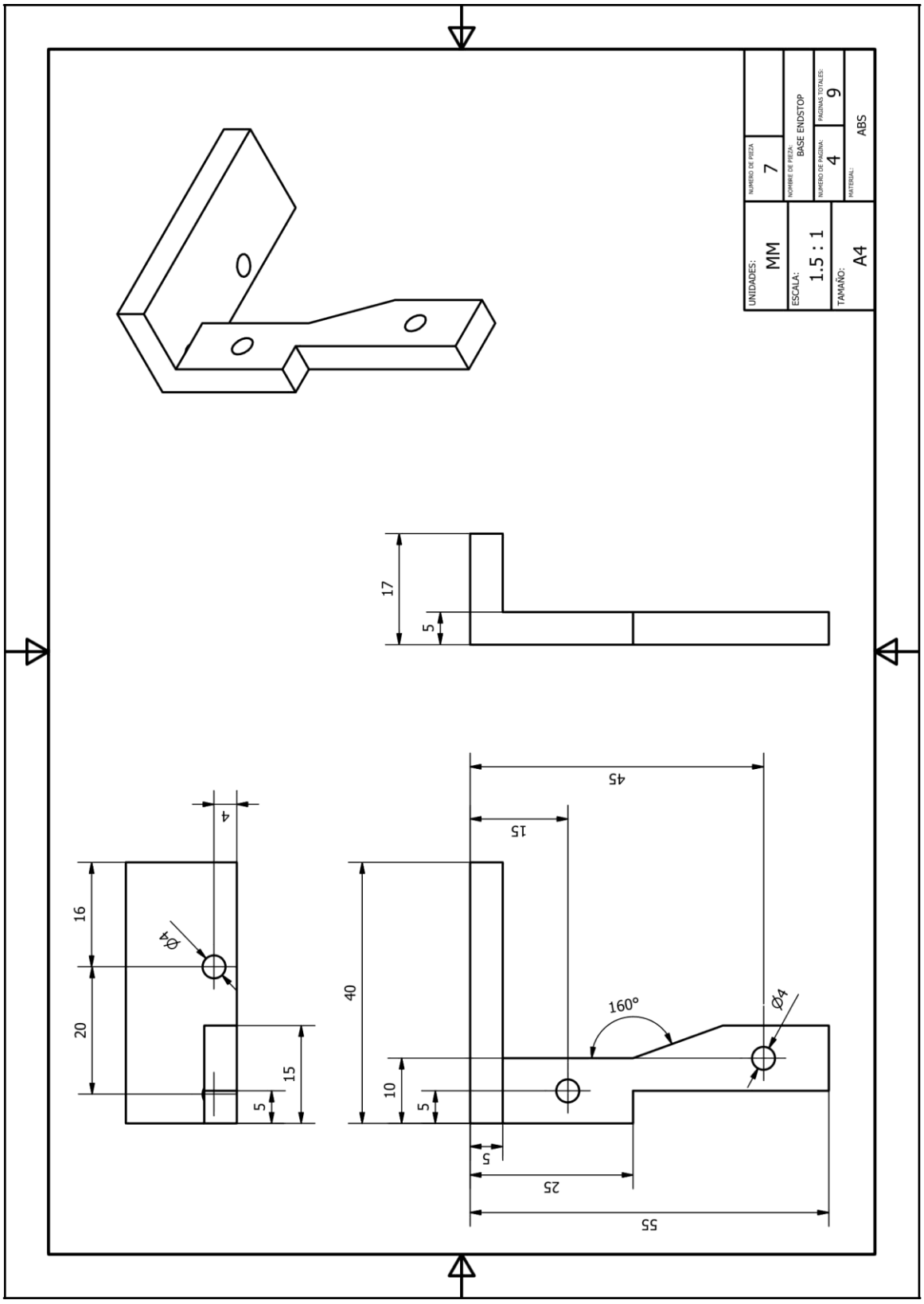

NÚMERO DE PIEZA: 19
NOMBRE DE PIEZA: PLATAFORMA MECANISMO GIRO
NÚMERO DE PÁGINA: 9
PÁGINAS TOTALES: 9
MATERIAL: ABS
UNIDADES: MM
ESCALA: 1 : 2
TAMAÑO: A4
136
130
ϕ3
ϕ4
R14
200
128
118
100
83
73
25
20
10
9
91
110
160
180
38
47
57
73
93
117

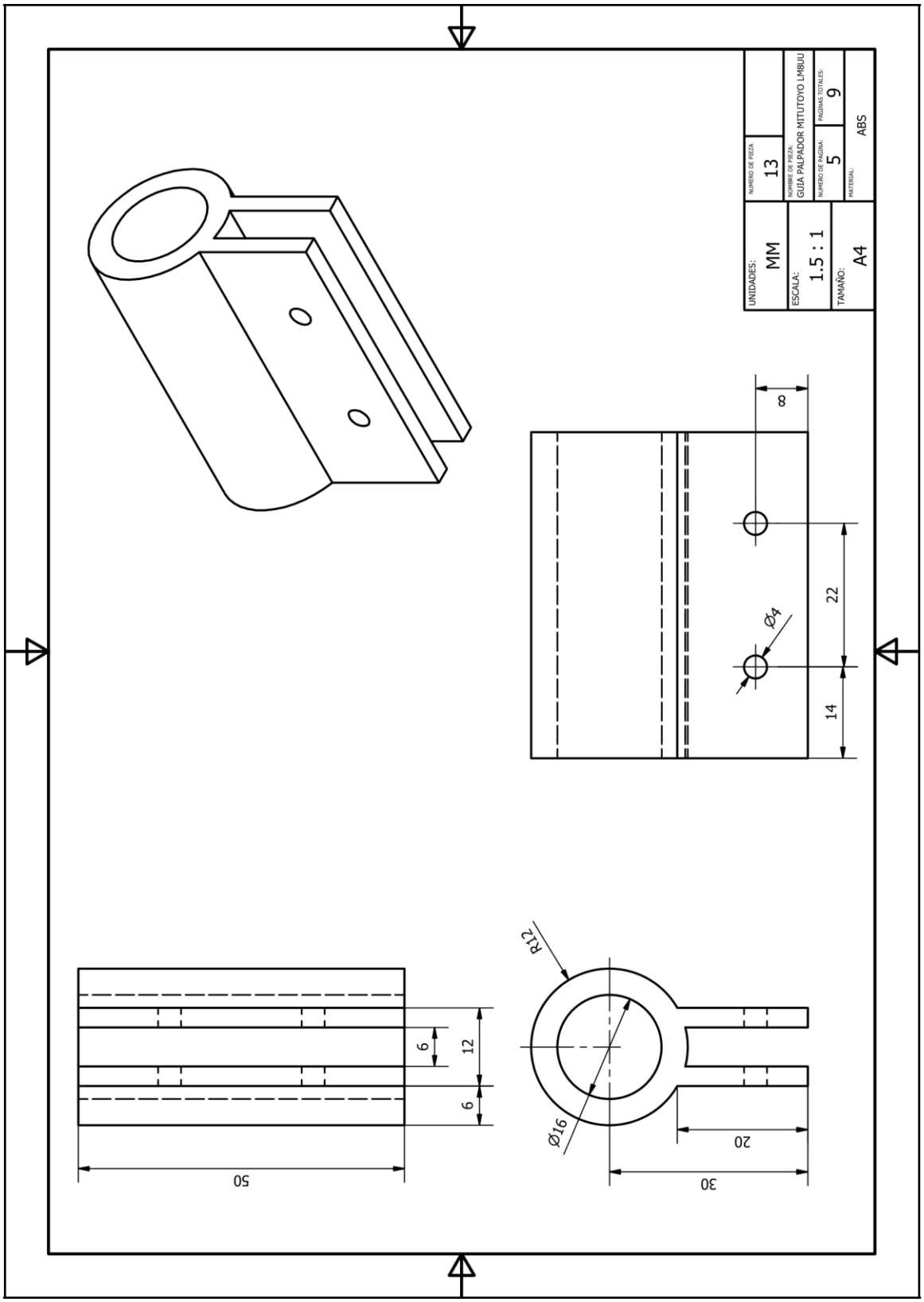
17.5. Planos mecanismo mitutoyo

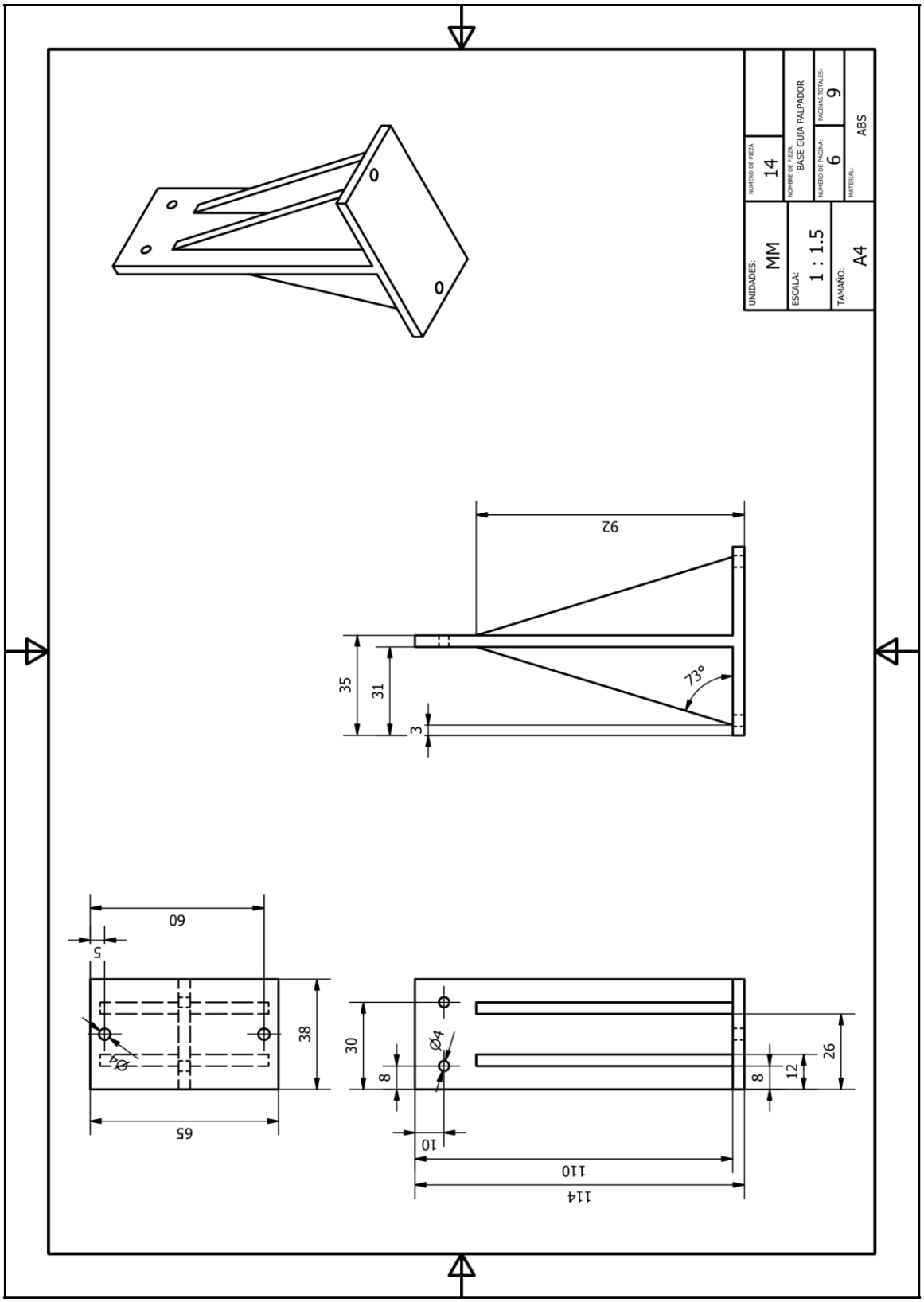


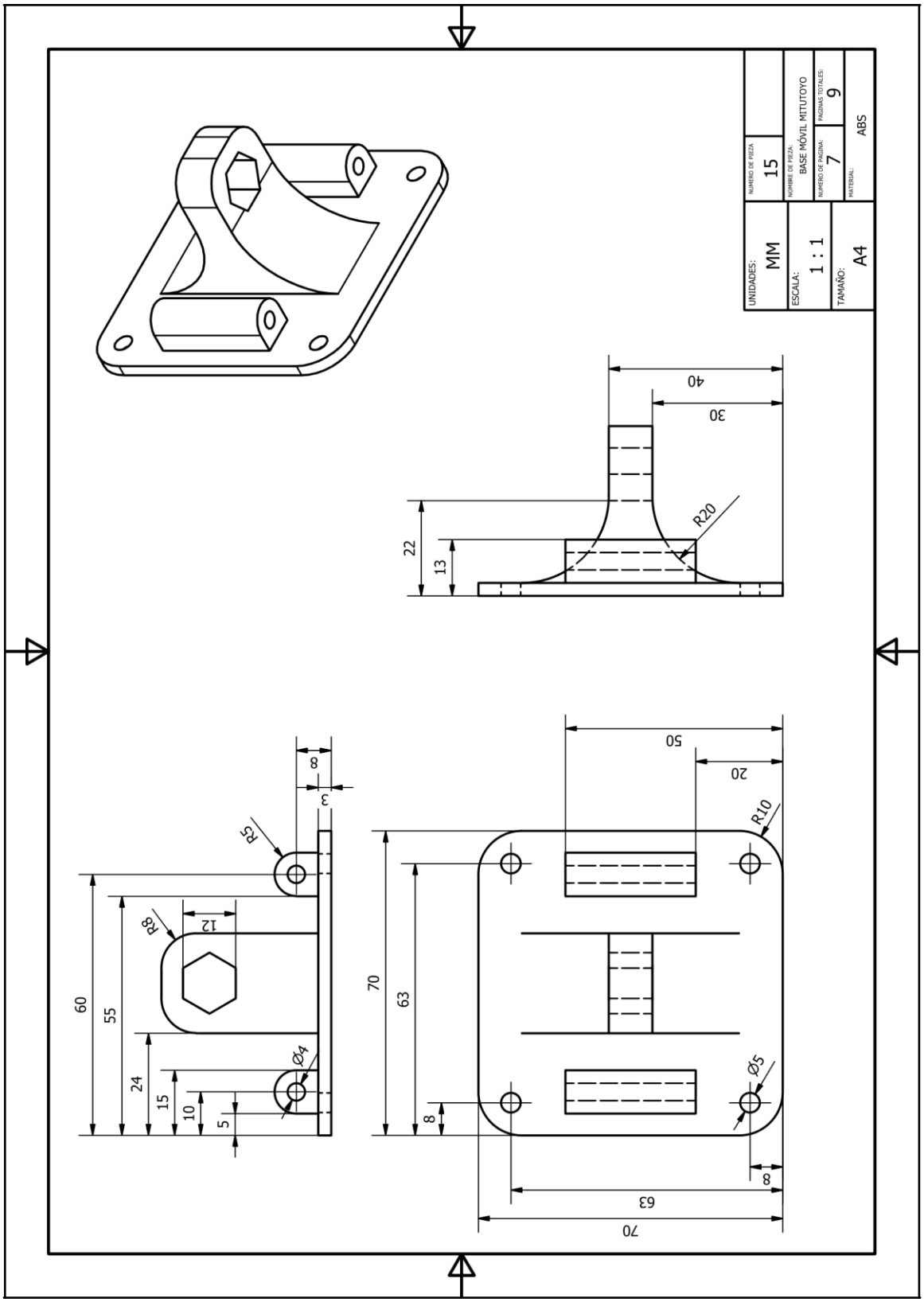

UNIDADES: MM
ESCALA: 1 : 1.5
TAMAÑO: A4
NUMERO DE PIEZA: 1
NOMBRE DE PIEZA: BASE STEPPER MITUTOYO
NUMERO DE PAGINA: 2
PAGINAS TOTALES: 9
MATERIAL: ABS
30
15
10
5
50
42
Ø25
Ø4
40
13
75
55
60
Ø5
15
20
35
27
110°
117

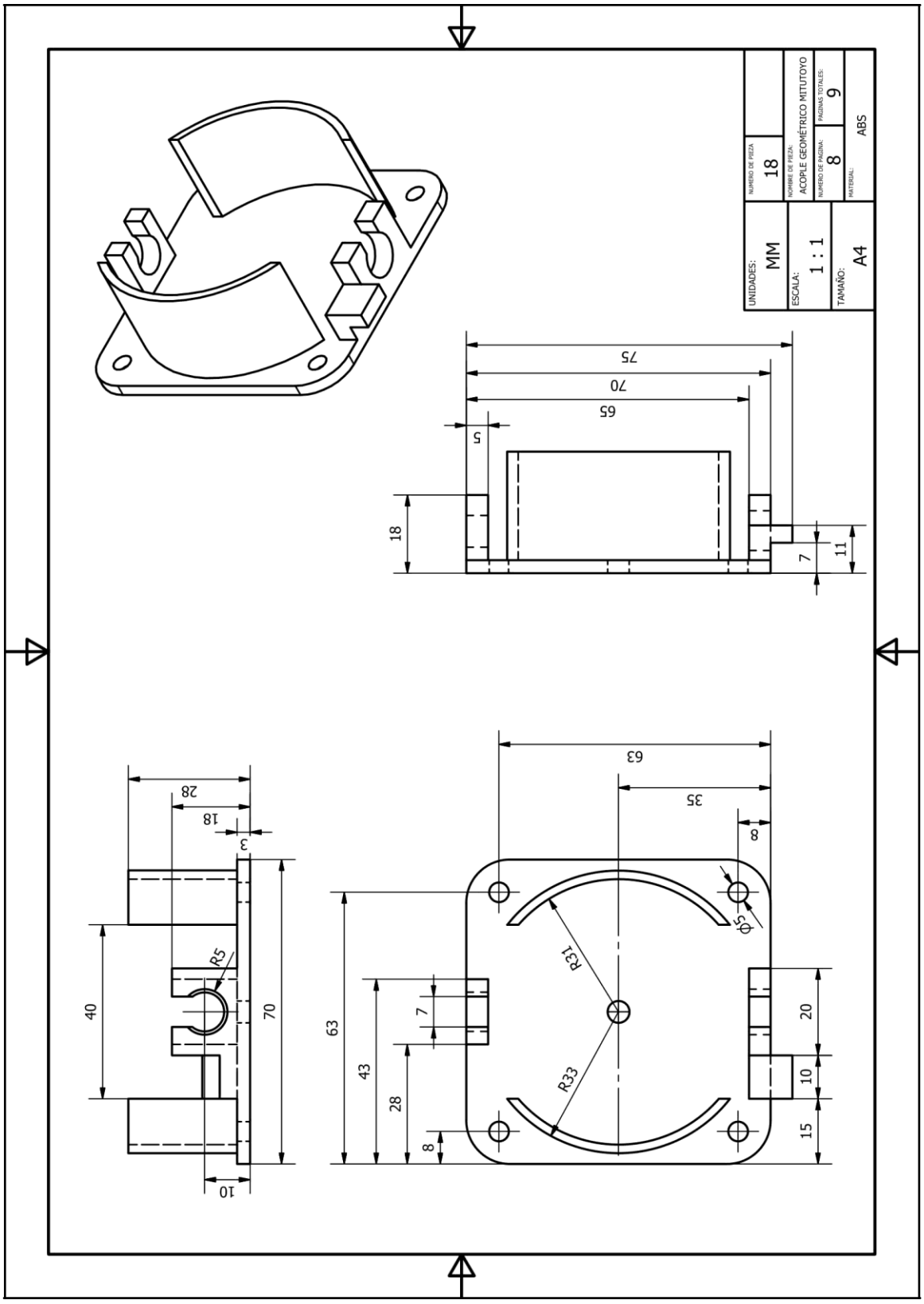

UNIDADES: MM
ESCALA: 1 : 1.5
TAMAÑO: A4
NUMERO DE PIEZA: 2
NOMBRE DE PIEZA: BASE RODAMIENTO MITUTOYO
NUMERO DE PAGINA: 3
PAGINAS TOTALES: 9
MATERIAL: ABS
5
50
45
55
13
Ø4
Ø17
57
62
83
103
35
20
Ø5
60
75
15
10
55
75
5
7
30
110°
117

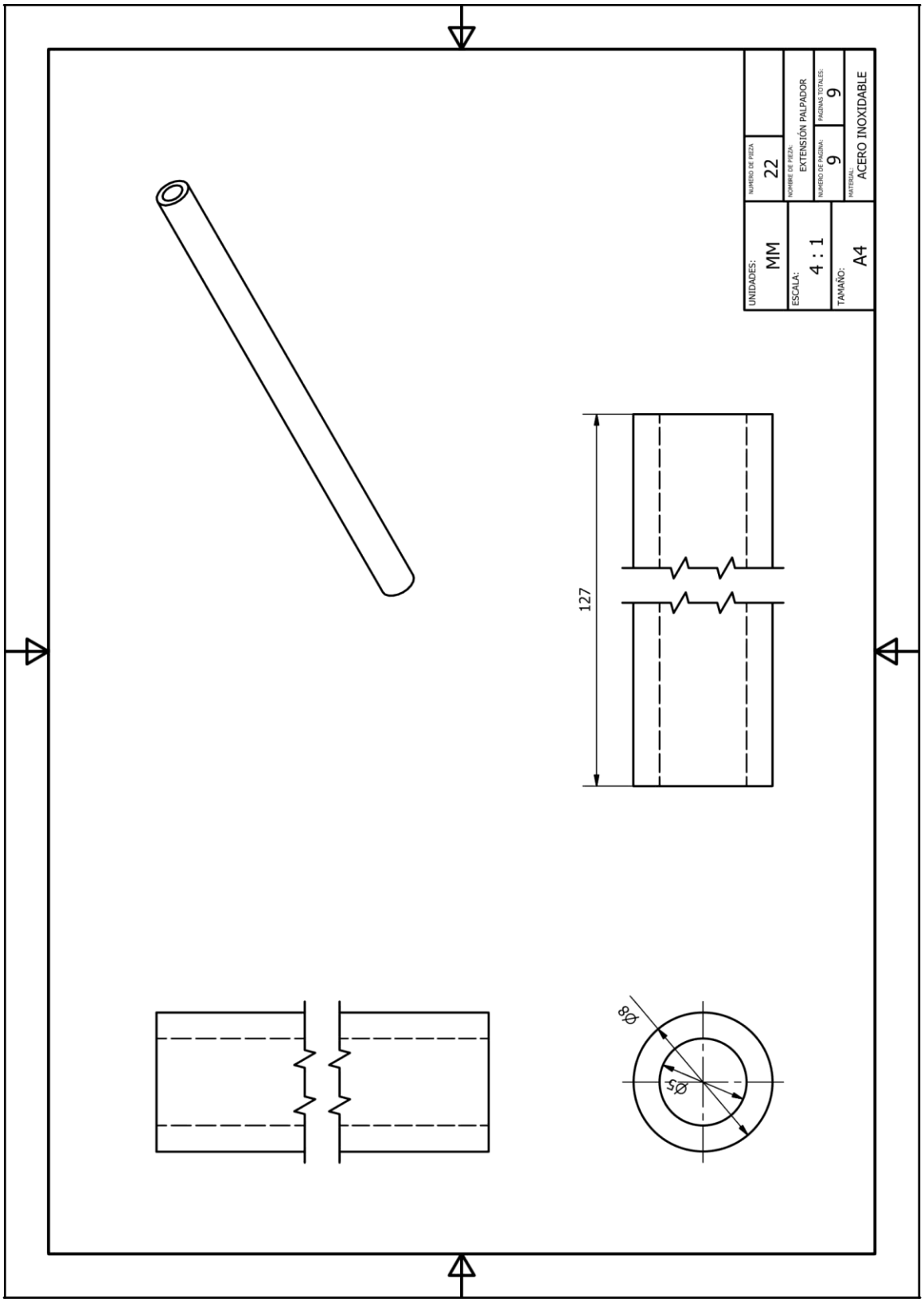

NÚMERO DE PIEZA:
7
NOMBRE DE PIEZA:
BASE ENDSTOP
NÚMERO DE PÁGINA:
4
PÁGINAS TOTALES:
9
MATERIAL:
ABS
UNIDADES:
MM
ESCALA:
1.5 : 1
TAMAÑO:
A4
17
5
45
15
40
160°
Ø4
10
5
5
25
55
4
16
Ø4
20
15
5


UNIDADES: MM
ESCALA: 1.5 : 1
TAMAÑO: A4
NUMERO DE PIEZA: 13
NOMBRE DE PIEZA: GUIA PALPADOR MITUTOYO LM8UU
NUMERO DE PAGINA: 5
PAGINAS TOTALES: 9
MATERIAL: ABS
8
22
14
Ø4
R12
Ø16
20
30
6
12
6
50

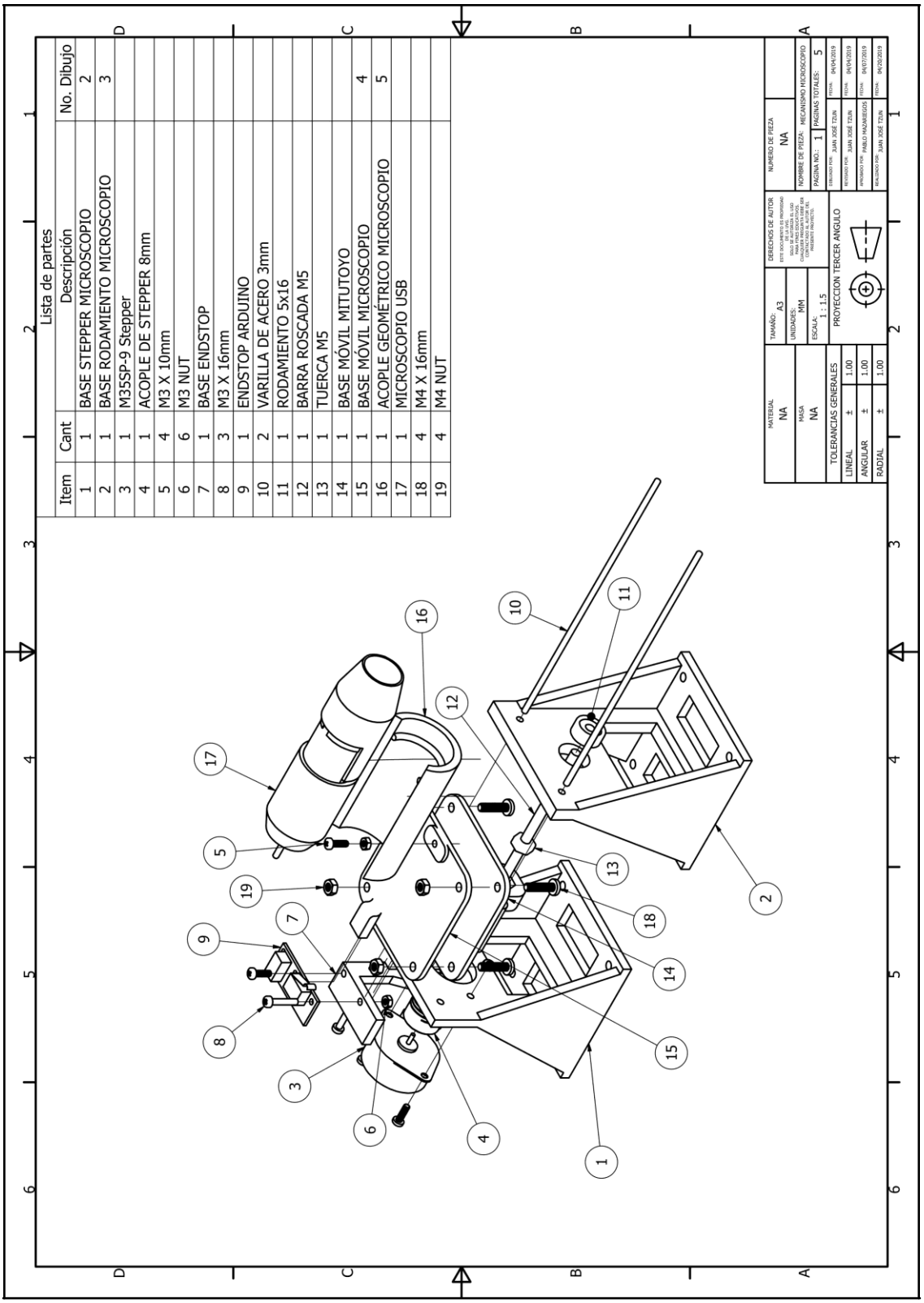

NUMERO DE PIEZA: 14
NOMBRE DE PIEZA: BASE GUIA PALPADOR
NUMERO DE PAGINA: 6
PAGINAS TOTALES: 9
MATERIAL: ABS
UNIDADES: MM
ESCALA: 1 : 1.5
TAMAÑO: A4
92
35
31
3
73°
60
5
38
65
Ø4
30
8
10
110
114
8
12
26

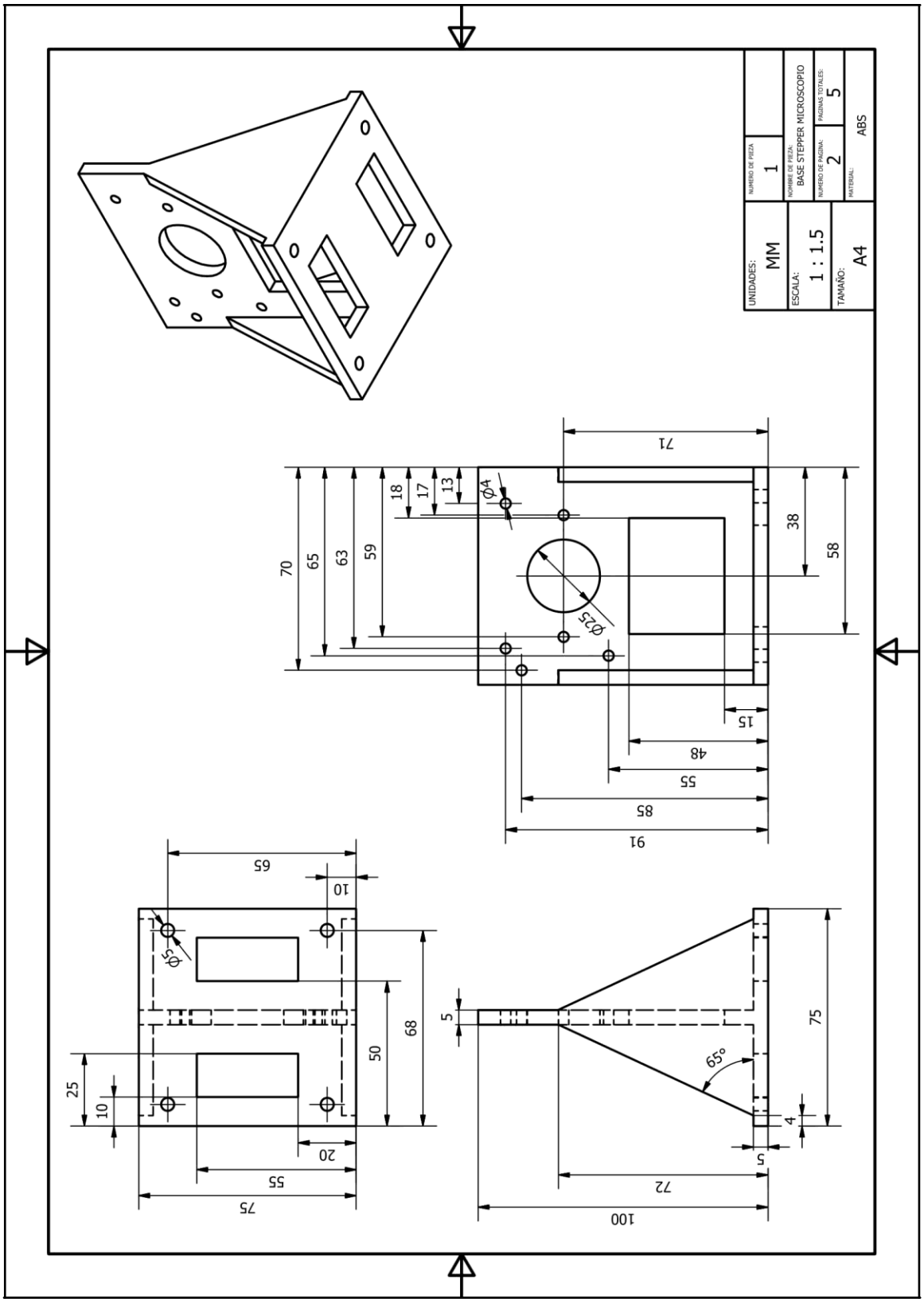

UNIDADES: MM
ESCALA: 1 : 1
TAMAÑO: A4
NUMERO DE PIEZA: 15
NOMBRE DE PIEZA: BASE MÓVIL MITUTOYO
NUMERO DE PAGINA: 7
PAGINAS TOTALES: 9
MATERIAL: ABS
40
30
R20
22
13
50
20
R10
70
63
8
Ø5
8
63
70
8
3
R5
R8
12
60
55
24
15
10
5
Ø4

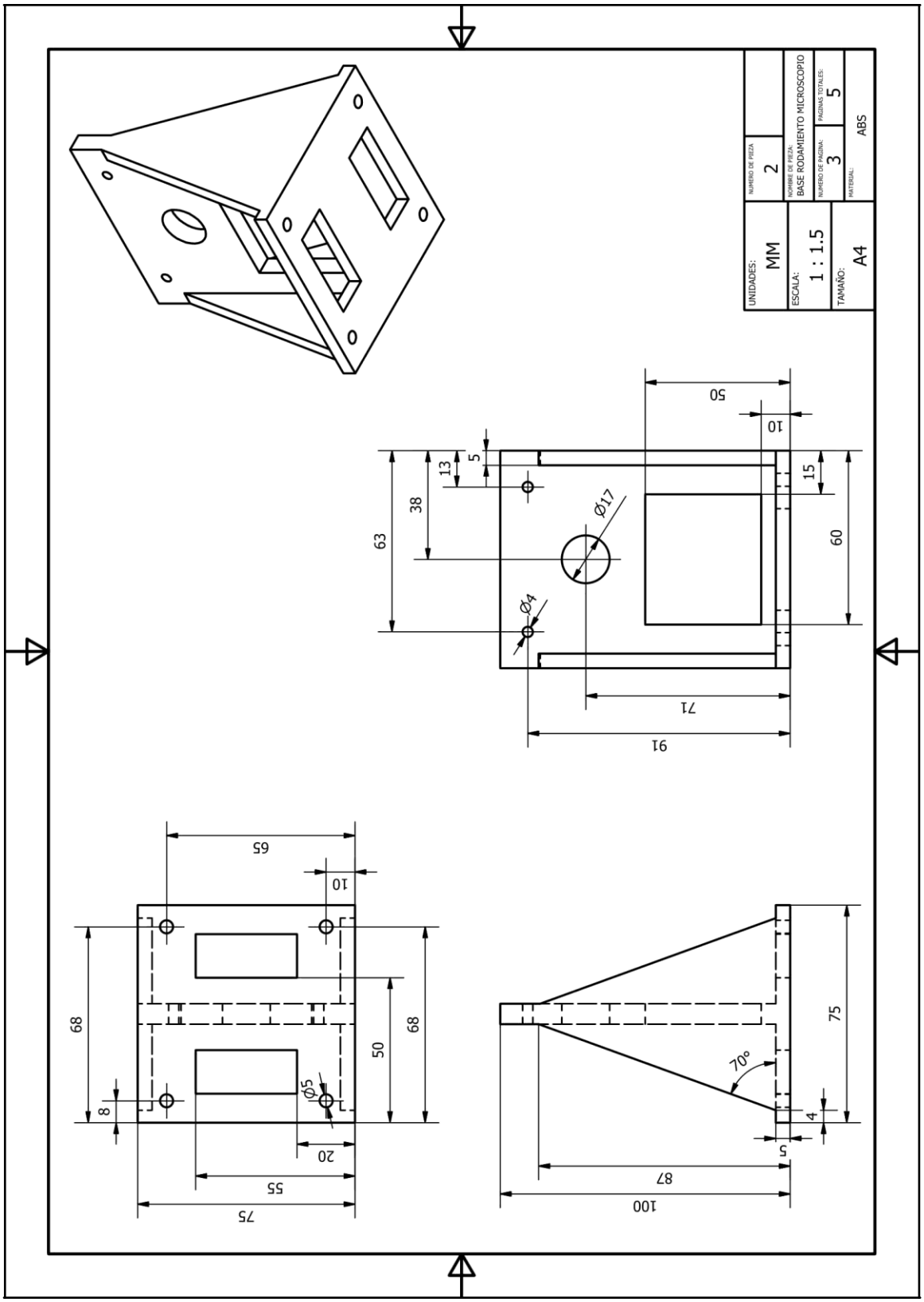

NÚMERO DE PIEZA: 18
NOMBRE DE PIEZA: ACOPLE GEOMÉTRICO MITUTOYO
NÚMERO DE PÁGINA: 8
PÁGINAS TOTALES: 9
MATERIAL: ABS
UNIDADES: MM
ESCALA: 1 : 1
TAMAÑO: A4
75
70
65
5
18
7
11
63
35
8
R31
R33
ϕ5
20
10
15
43
28
7
28
18
3
R5
40
70
10

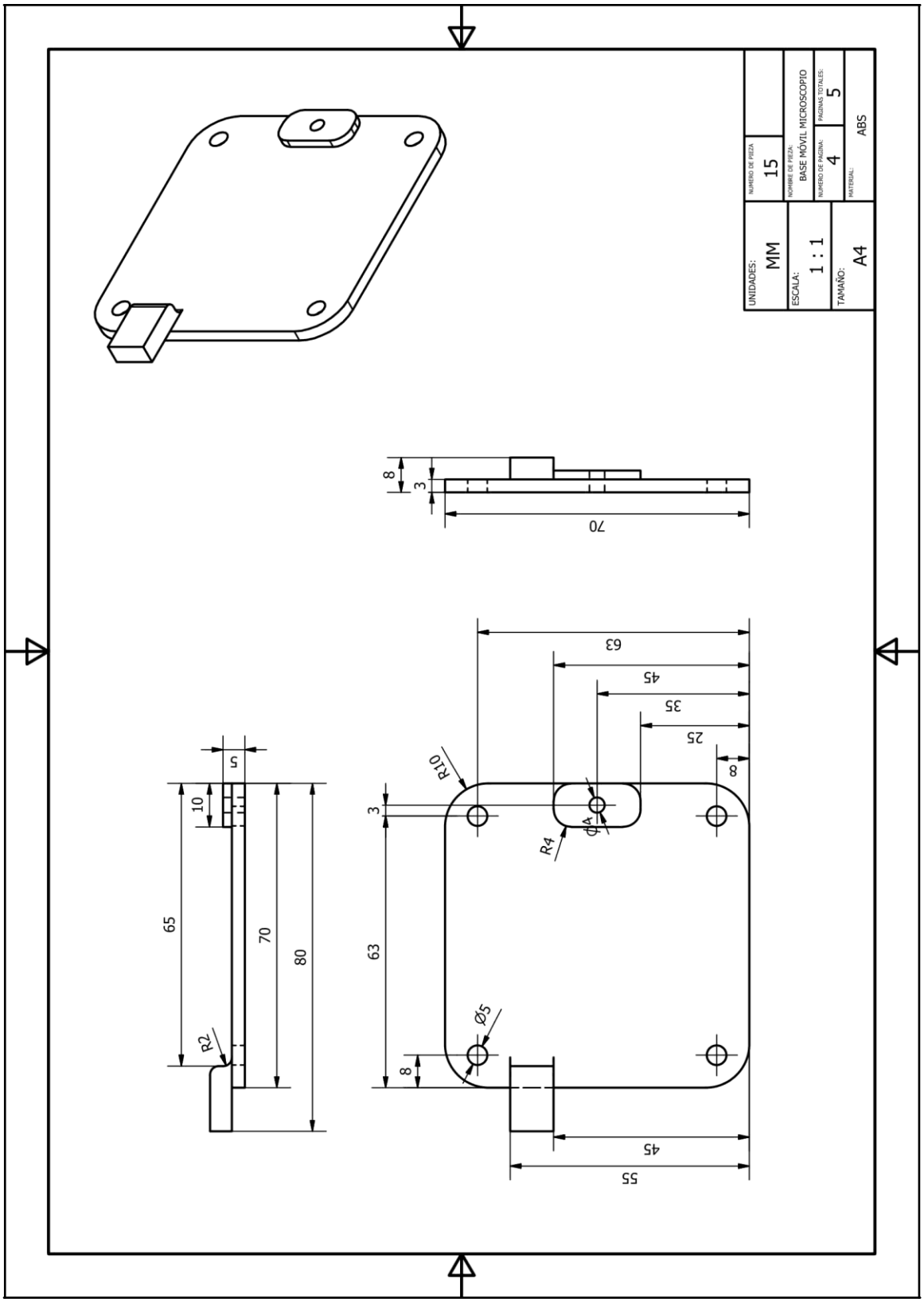

UNIDADES: MM
ESCALA: 4 : 1
TAMAÑO: A4
NUMERO DE PIEZA: 22
NOMBRE DE PIEZA: EXTENSIÓN PALPADOR
NUMERO DE PAGINA: 9
PAGINAS TOTALES: 9
MATERIAL: ACERO INOXIDABLE
127
8Ø
5Ø

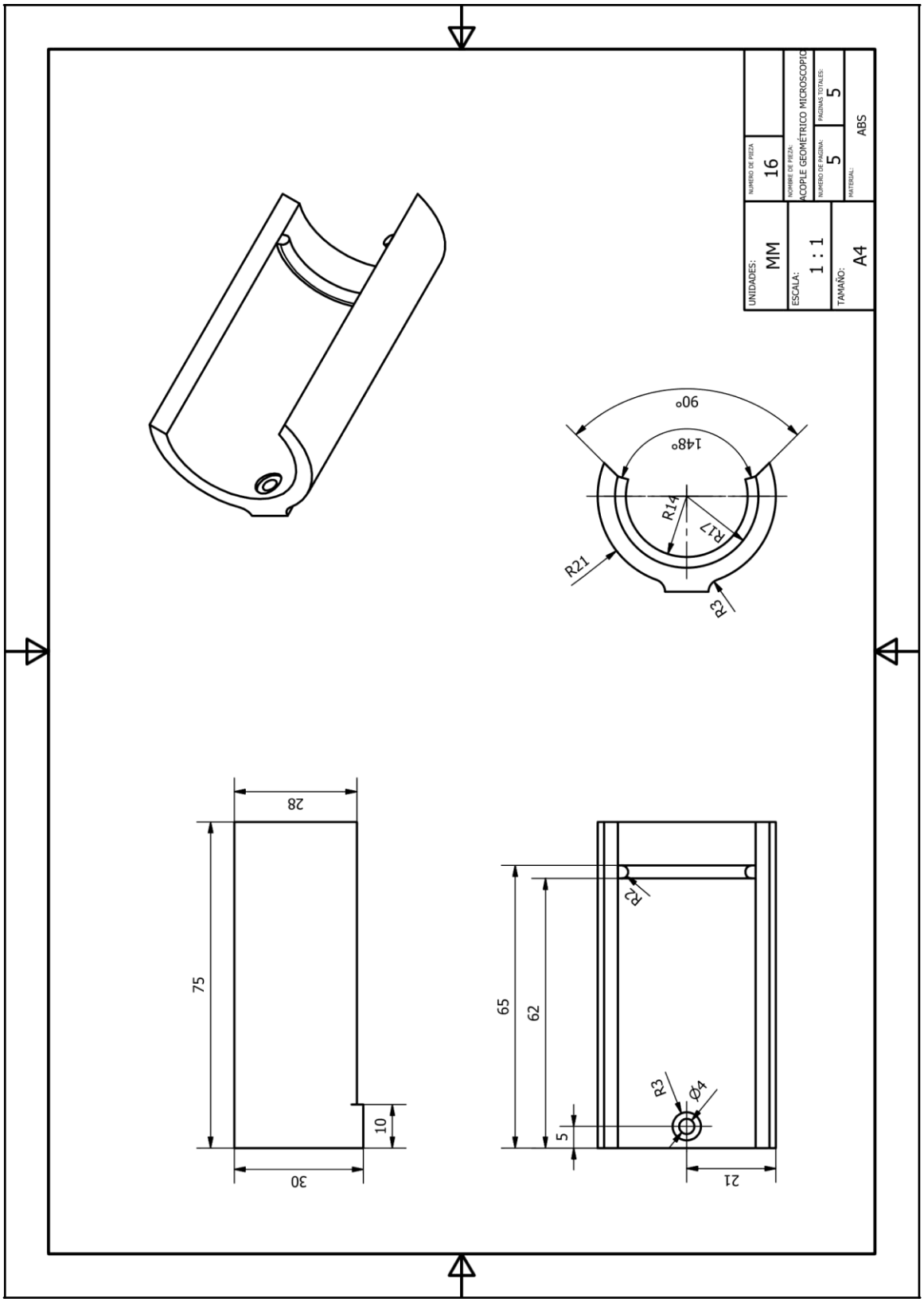
17.6. Planos mecanismo microscopio



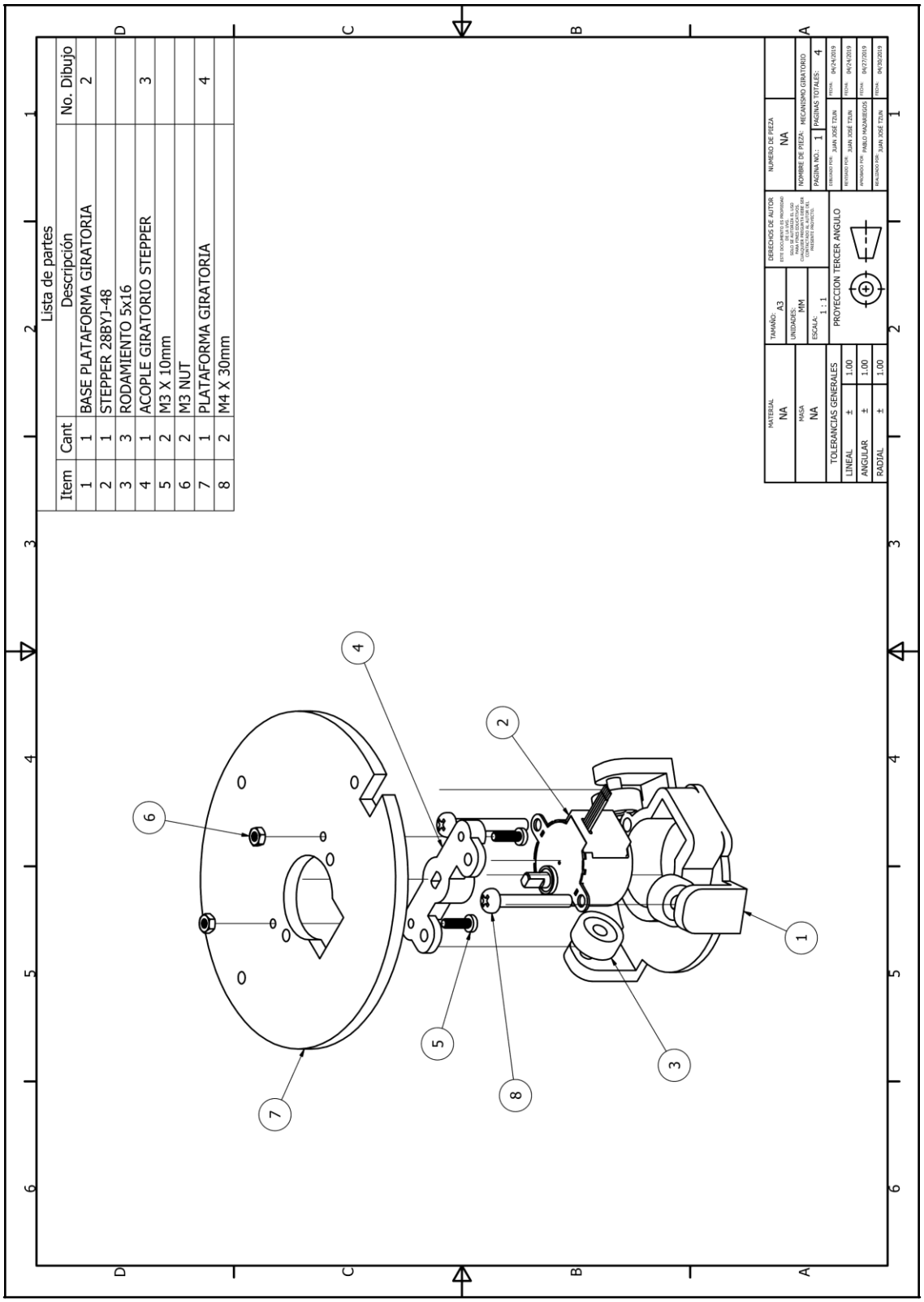

NUMERO DE PIEZA: 1
NOMBRE DE PIEZA: BASE STEPPER MICROSCOPIO
NUMERO DE PAGINA: 2
PAGINAS TOTALES: 5
MATERIAL: ABS
UNIDADES: MM
ESCALA: 1 : 1.5
TAMAÑO: A4
71
70
65
63
59
18
17
13
ΦA
38
58
Φ25
15
48
55
85
91
65
10
Φ5
68
50
25
10
20
55
75
5
65°
75
4
5
72
100

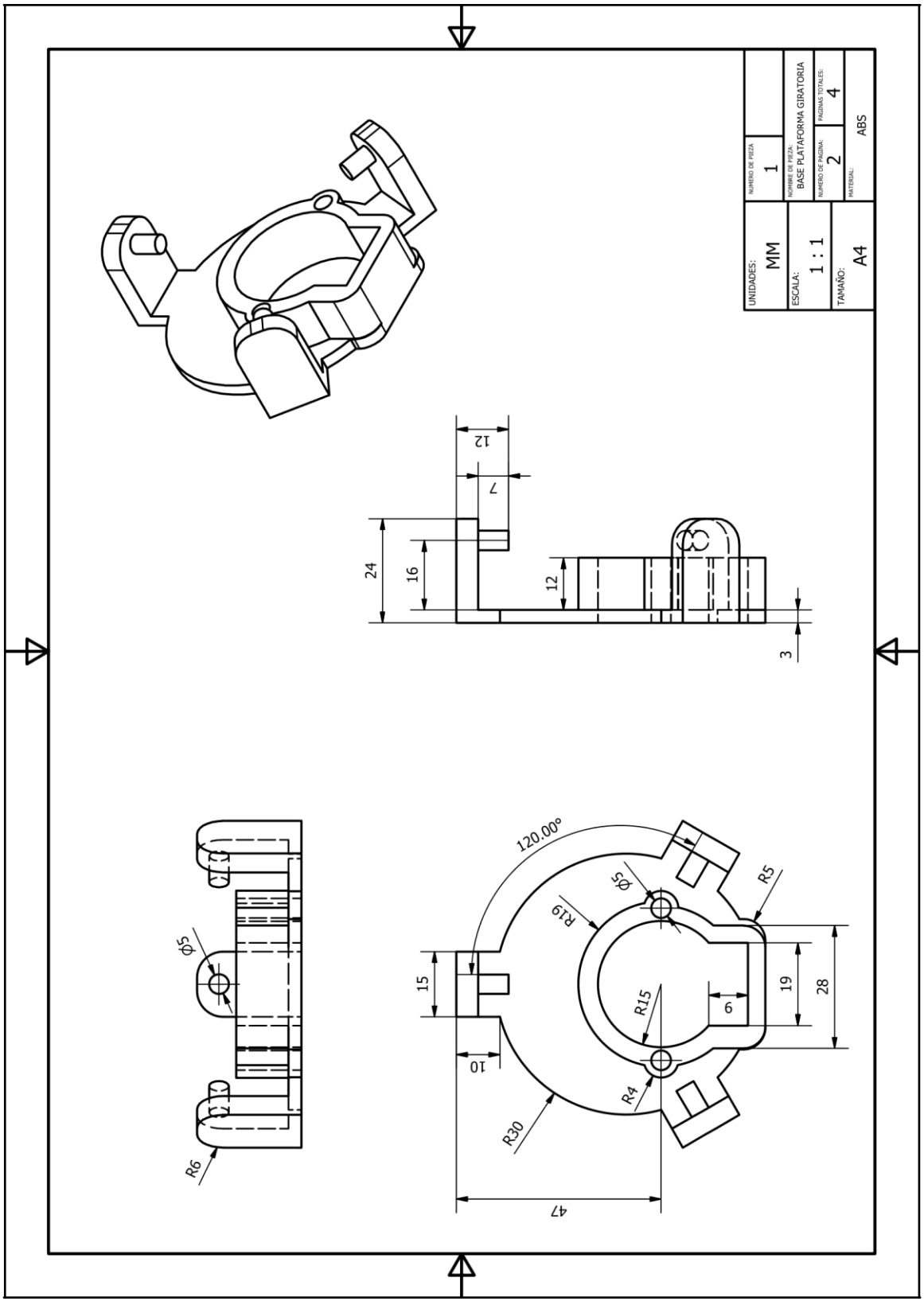

NUMERO DE PIEZA: 2
NOMBRE DE PIEZA: BASE RODAMIENTO MICROSCOPIO
NUMERO DE PAGINA: 3
PAGINAS TOTALES: 5
MATERIAL: ABS
UNIDADES: MM
ESCALA: 1 : 1.5
TAMAÑO: A4
50
10
13
5
15
38
63
60
Ø17
Ø4
71
91
65
10
68
68
50
8
Ø5
20
55
75
75
70°
4
5
87
100

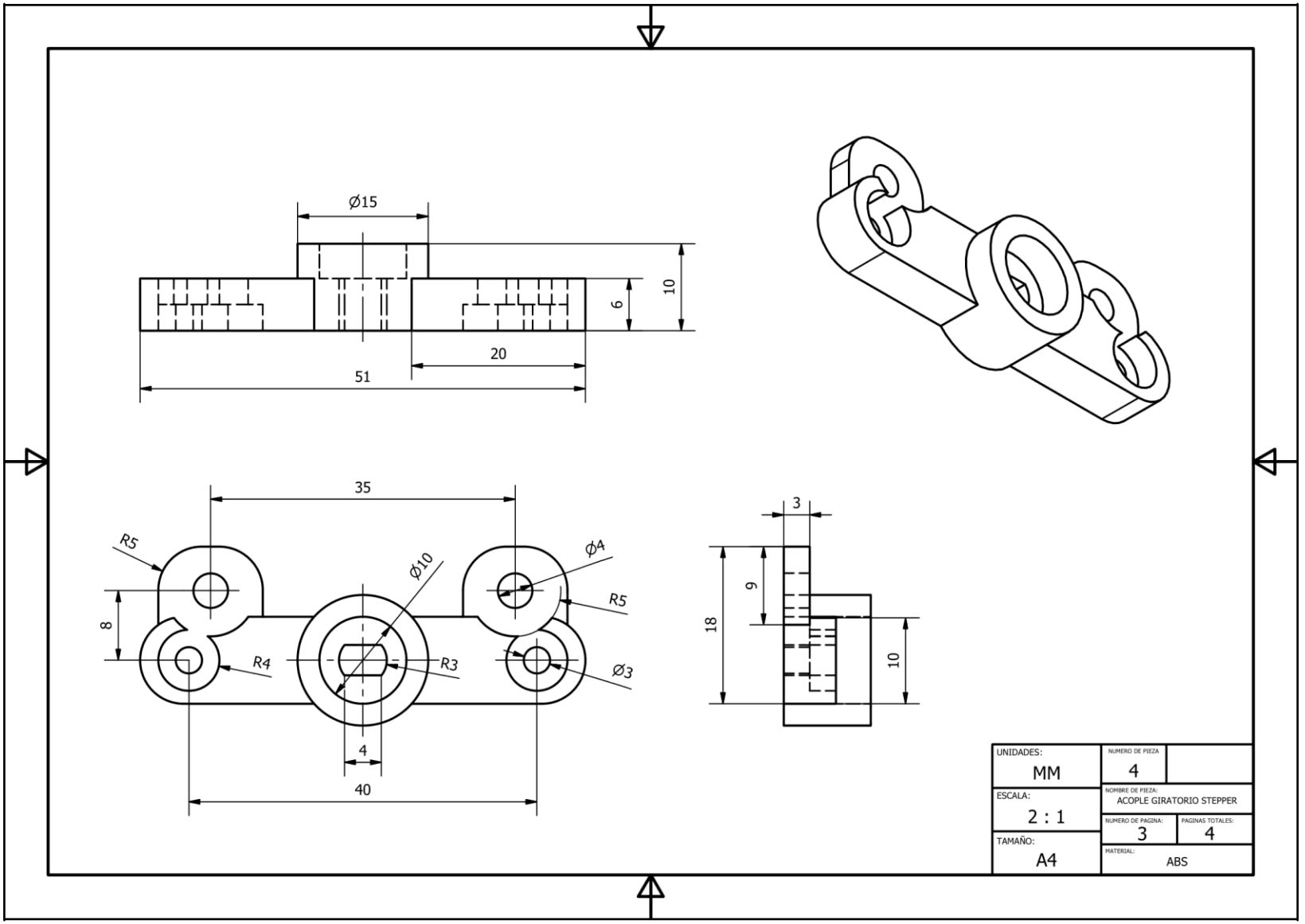

NUMERO DE PIEZA: 15
NOMBRE DE PIEZA: BASE MÓVIL MICROSCOPIO
NUMERO DE PAGINA: 4
PAGINAS TOTALES: 5
MATERIAL: ABS
UNIDADES: MM
ESCALA: 1 : 1
TAMAÑO: A4
8
3
70
63
45
35
25
8
R10
3
Ø4
R4
63
8
Ø5
45
55
5
10
65
70
80
R2

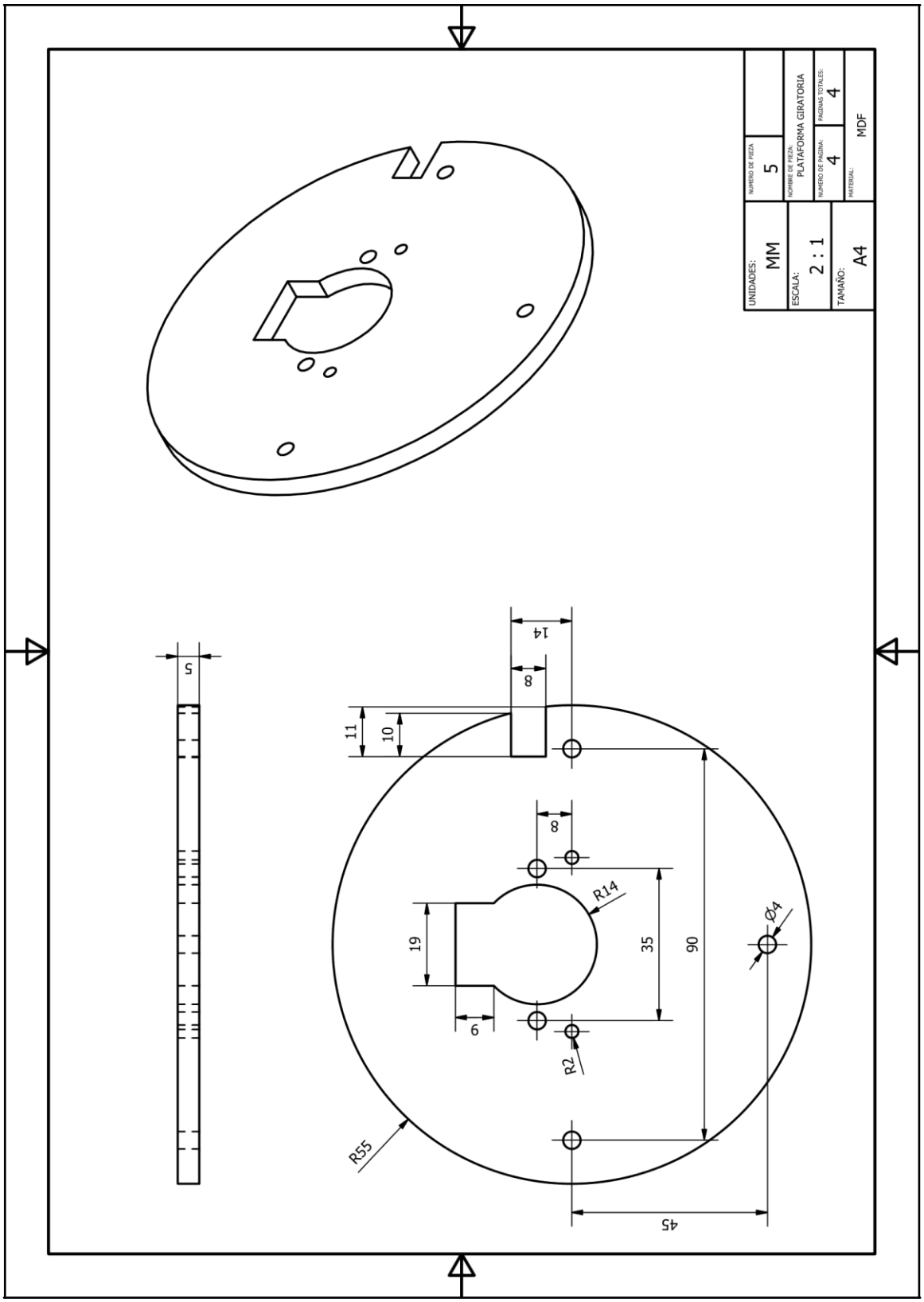

NÚMERO DE PIEZA: 16
NOMBRE DE PIEZA: ACOPLE GEOMÉTRICO MICROSCOPIO
NÚMERO DE PÁGINA: 5
PÁGINAS TOTALES: 5
MATERIAL: ABS
UNIDADES: MM
ESCALA: 1 : 1
TAMAÑO: A4
90°
148°
R14
R17
R21
R3
28
75
10
30
65
62
5
R2
R3
Ø4
21

17.7. Planos mecanismo giratorio

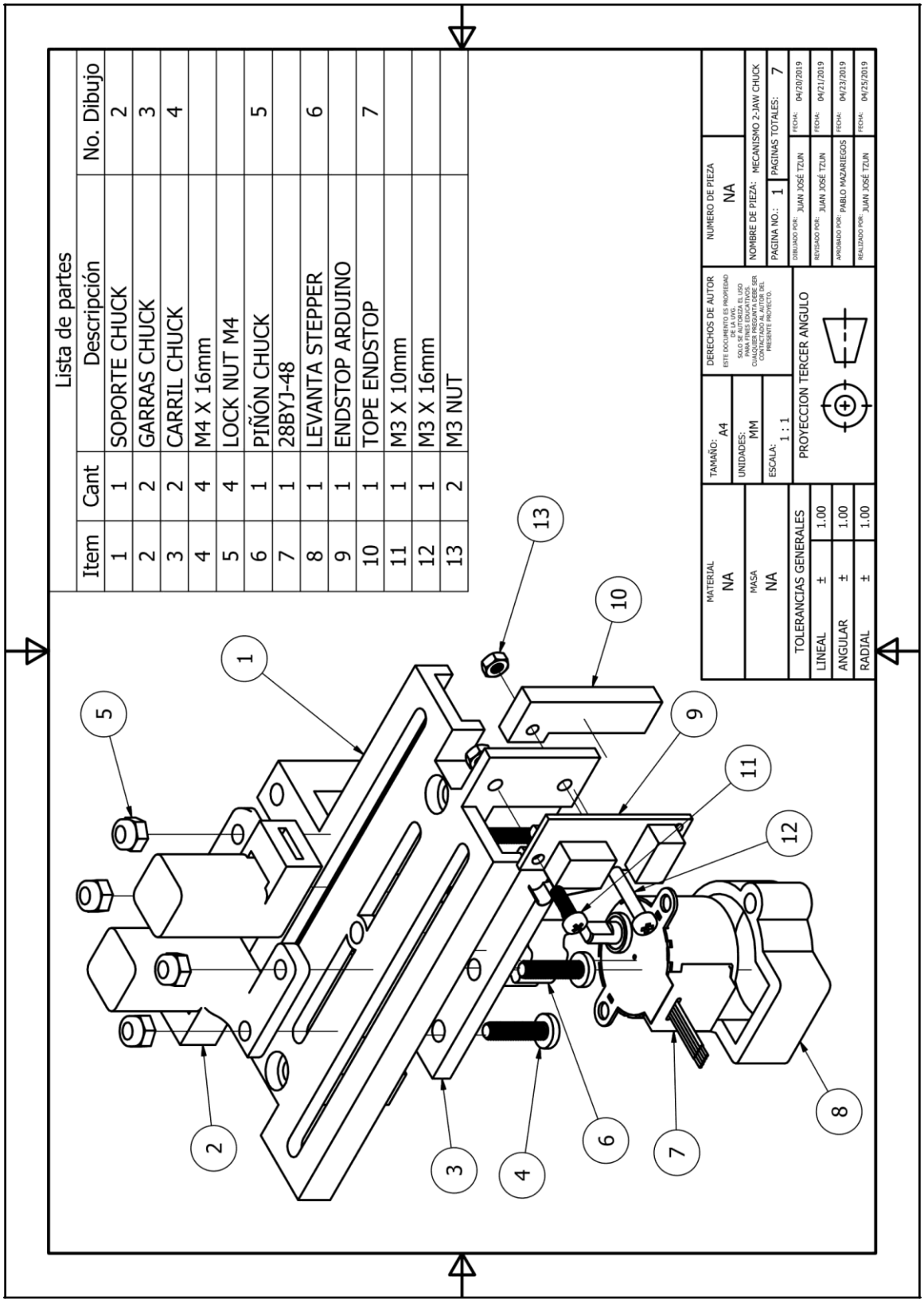


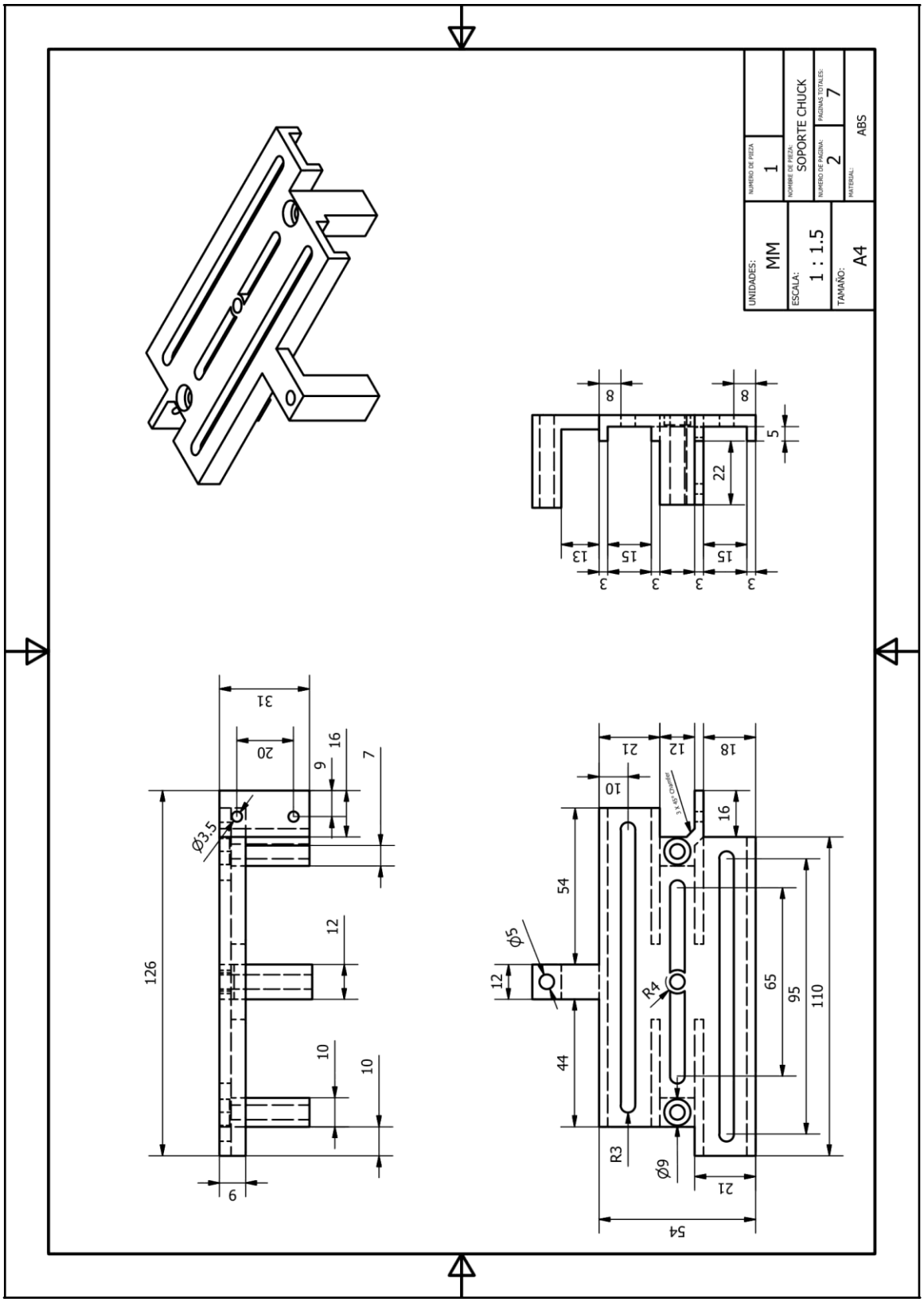

UNIDADES: MM
ESCALA: 1 : 1
TAMAÑO: A4
NUMERO DE PIEZA: 1
NOMBRE DE PIEZA: BASE PLATAFORMA GIRATORIA
NUMERO DE PAGINA: 2
PAGINAS TOTALES: 4
MATERIAL: ABS
12
7
24
16
12
3
120.00°
Ø5
R5
R19
15
19
28
9
R15
10
R4
R30
47
Ø5
R6

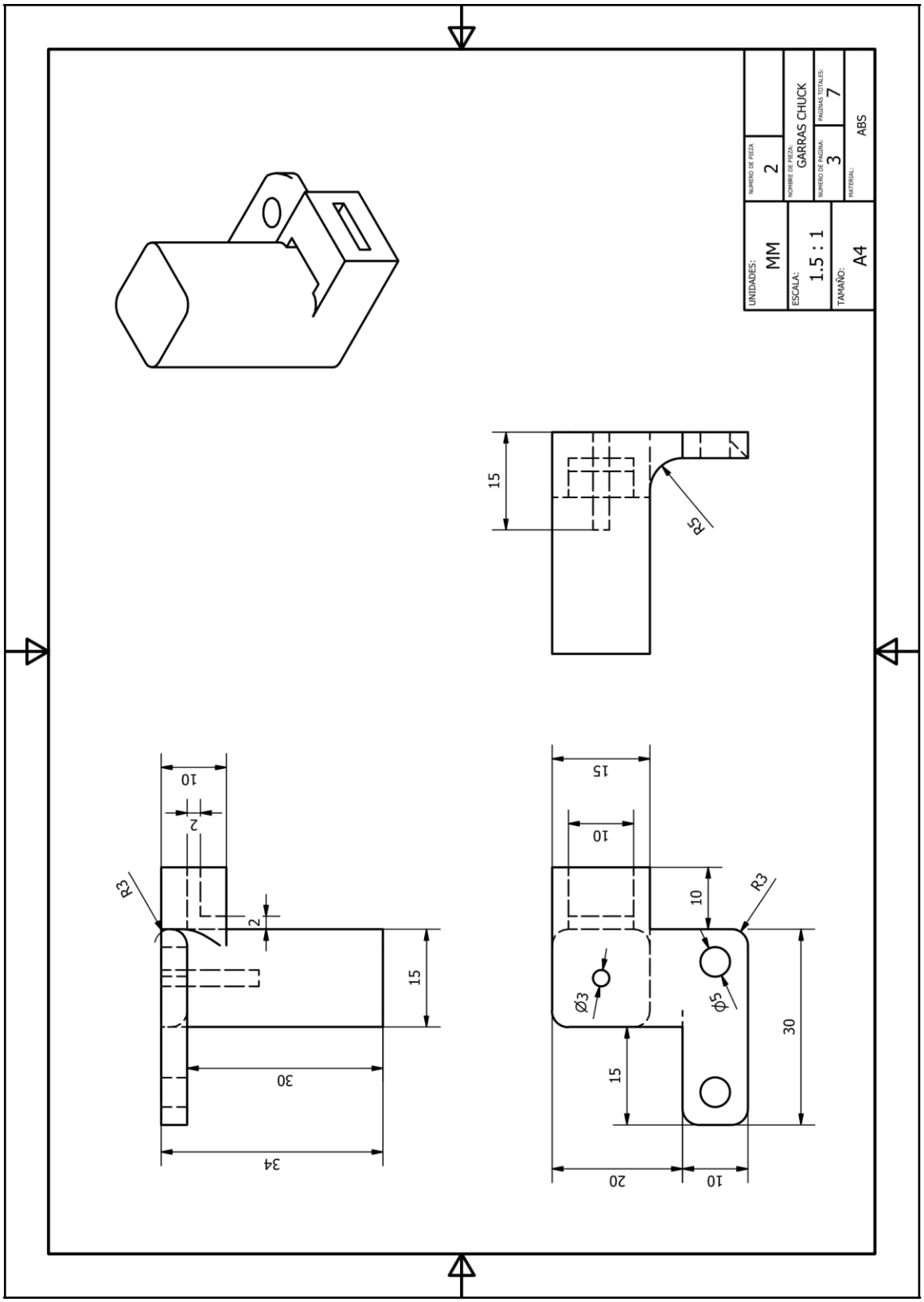

Ø15
10
6
20
51
35
R5
Ø10
Ø4
R5
8
R4
R3
Ø3
4
40
3
9
18
10
UNIDADES:
MM
NUMERO DE PIEZA
4
ESCALA:
2 : 1
NOMBRE DE PIEZA:
ACOPLE GIRATORIO STEPPER
NUMERO DE PAGINA:
3
PAGINAS TOTALES:
4
TAMAÑO:
A4
MATERIAL:
ABS

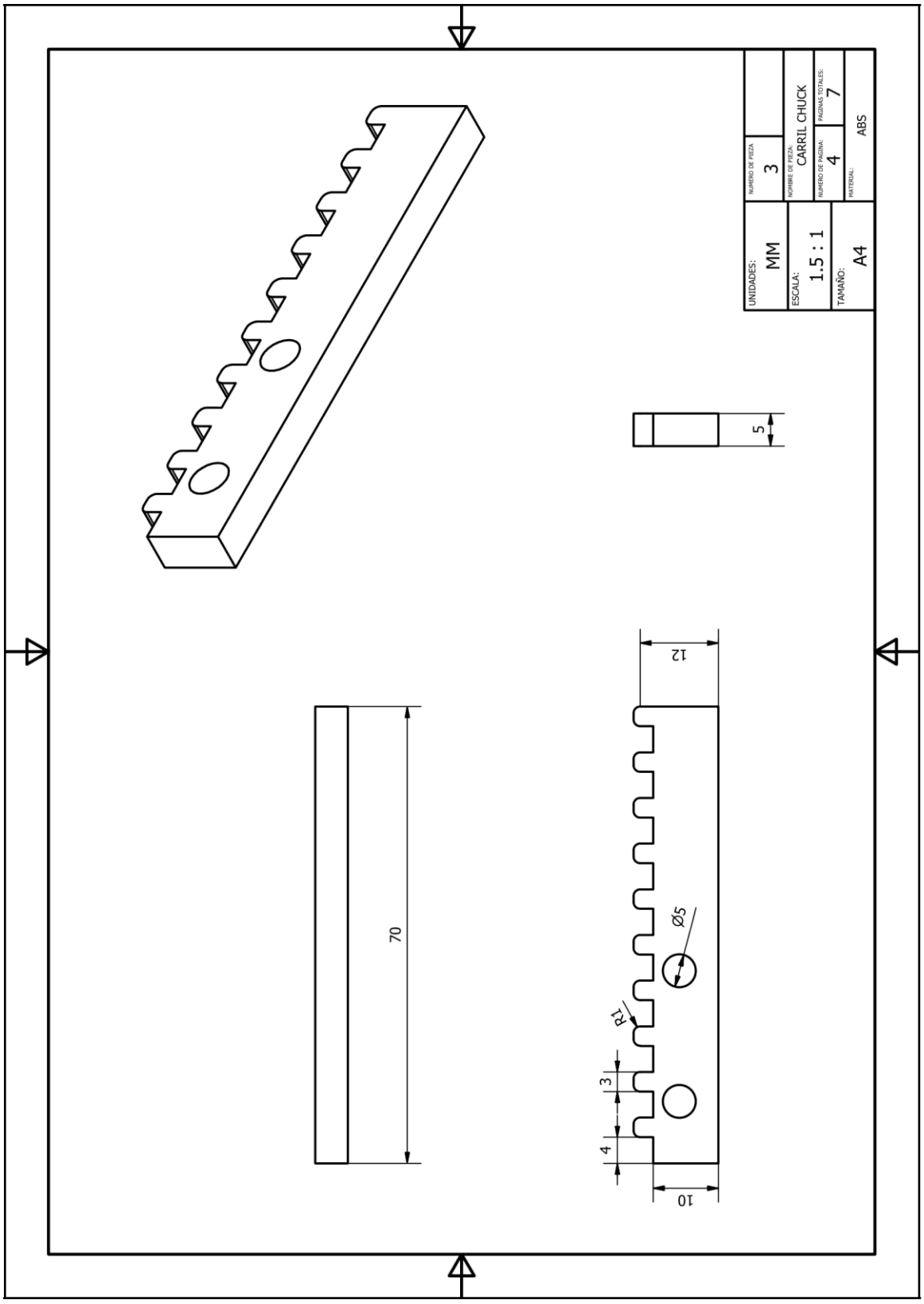

UNIDADES:
MM
ESCALA:
2 : 1
TAMAÑO:
A4
NUMERO DE PIEZA:
5
NOMBRE DE PIEZA:
PLATAFORMA GIRATORIA
NUMERO DE PAGINA:
4
PAGINAS TOTALES:
4
MATERIAL:
MDF
5
14
8
11
10
8
R14
19
35
90
Ø4
9
R2
R55
45

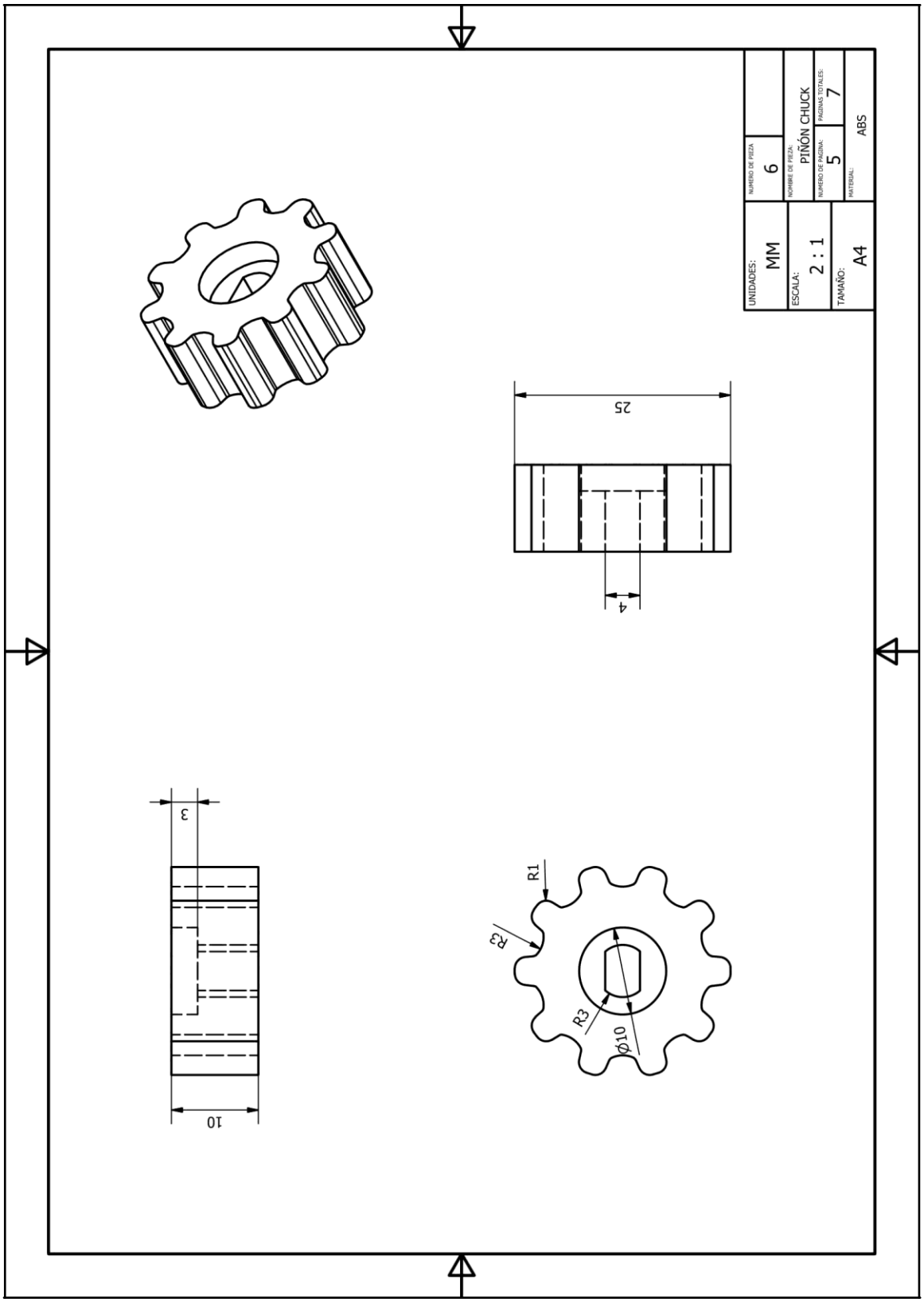
17.8. Planos mecanismo 2-Jaw Chuck

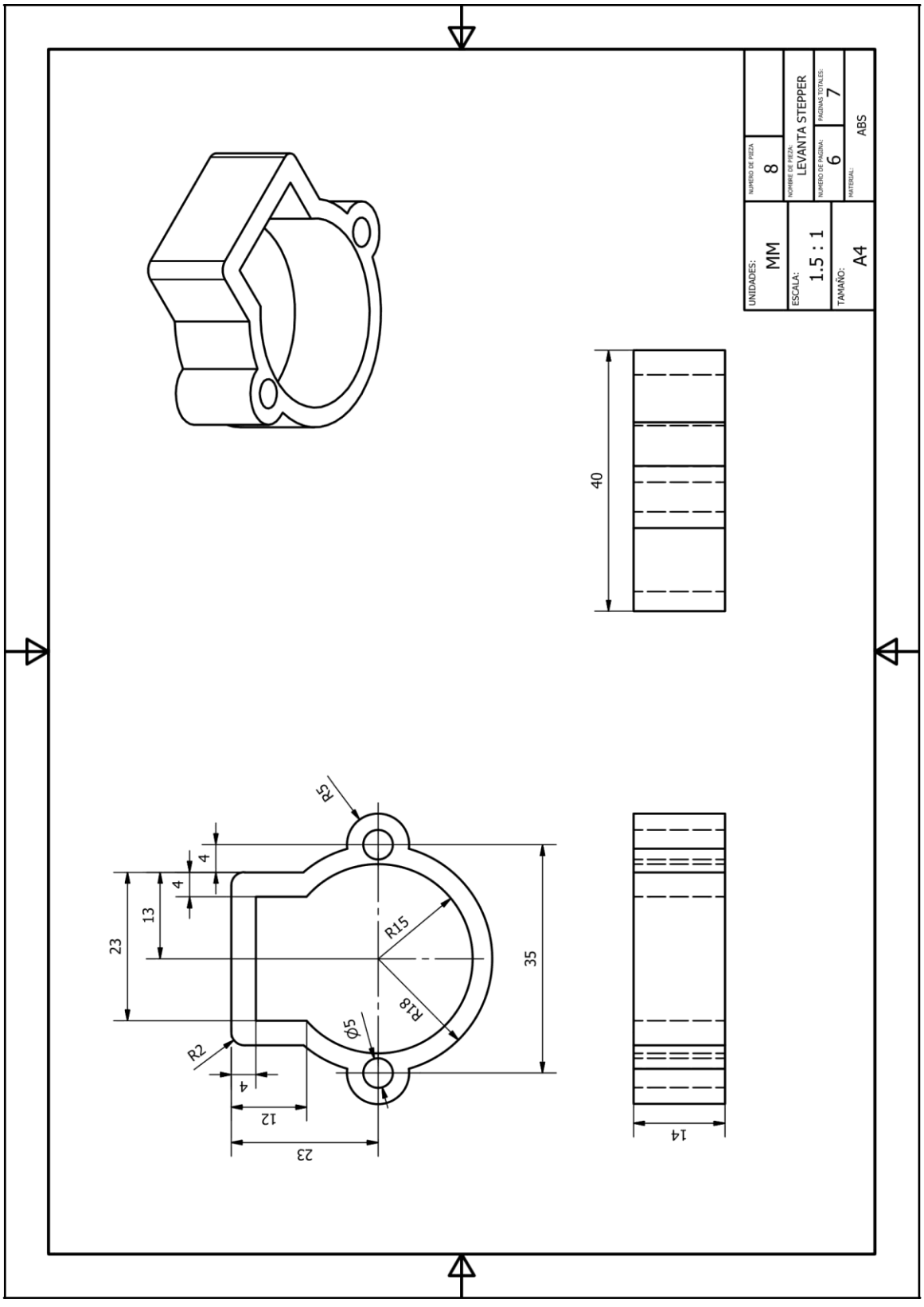


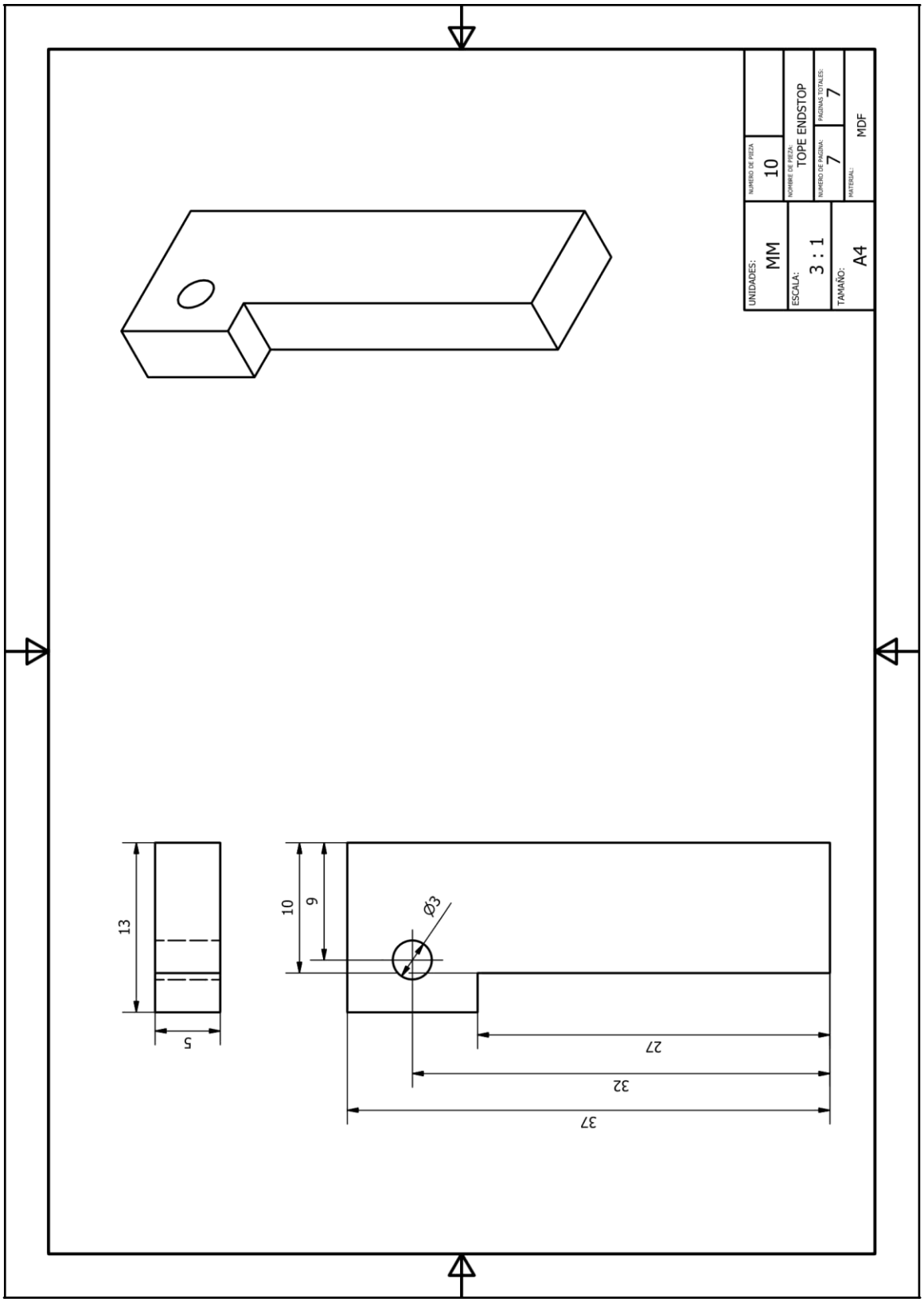

UNIDADES: MM
ESCALA: 1 : 1.5
TAMAÑO: A4
NUMERO DE PIEZA: 1
NOMBRE DE PIEZA: SOPORTE CHUCK
NUMERO DE PAGINA: 2
PAGINAS TOTALES: 7
MATERIAL: ABS
8
8
5
22
13
15
15
3
3
3
3
31
20
16
9
7
Ø3.5
126
12
10
10
6
21
12
18
10
16
3 X 45° Chamfer
54
Ø5
12
R4
44
65
95
110
R3
Ø9
21
54


UNIDADES: MM
ESCALA: 1.5 : 1
TAMAÑO: A4
NUMERO DE PIEZA: 2
NOMBRE DE PIEZA: GARRAS CHUCK
NUMERO DE PAGINA: 3
PAGINAS TOTALES: 7
MATERIAL: ABS
15
R2
10
2
R3
2
15
30
34
15
10
10
R3
Ø3
Ø5
30
15
20
10

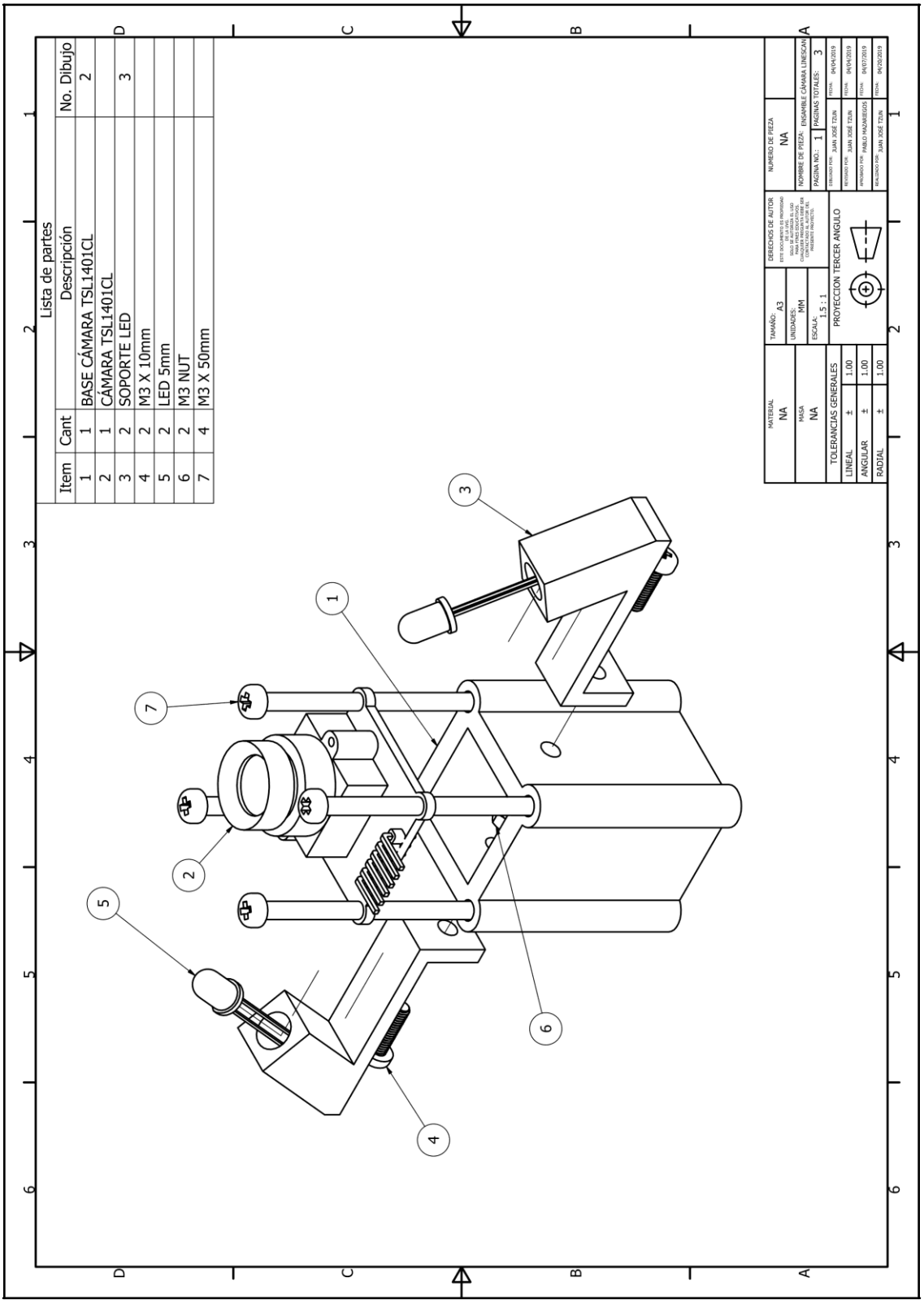

UNIDADES:
MM
ESCALA:
1.5 : 1
TAMAÑO:
A4
NUMERO DE PIEZA:
3
NOMBRE DE PIEZA:
CARRIL CHUCK
NUMERO DE PAGINA:
4
PAGINAS TOTALES:
7
MATERIAL:
ABS
5
70
12
Ø5
R1
3
4
10

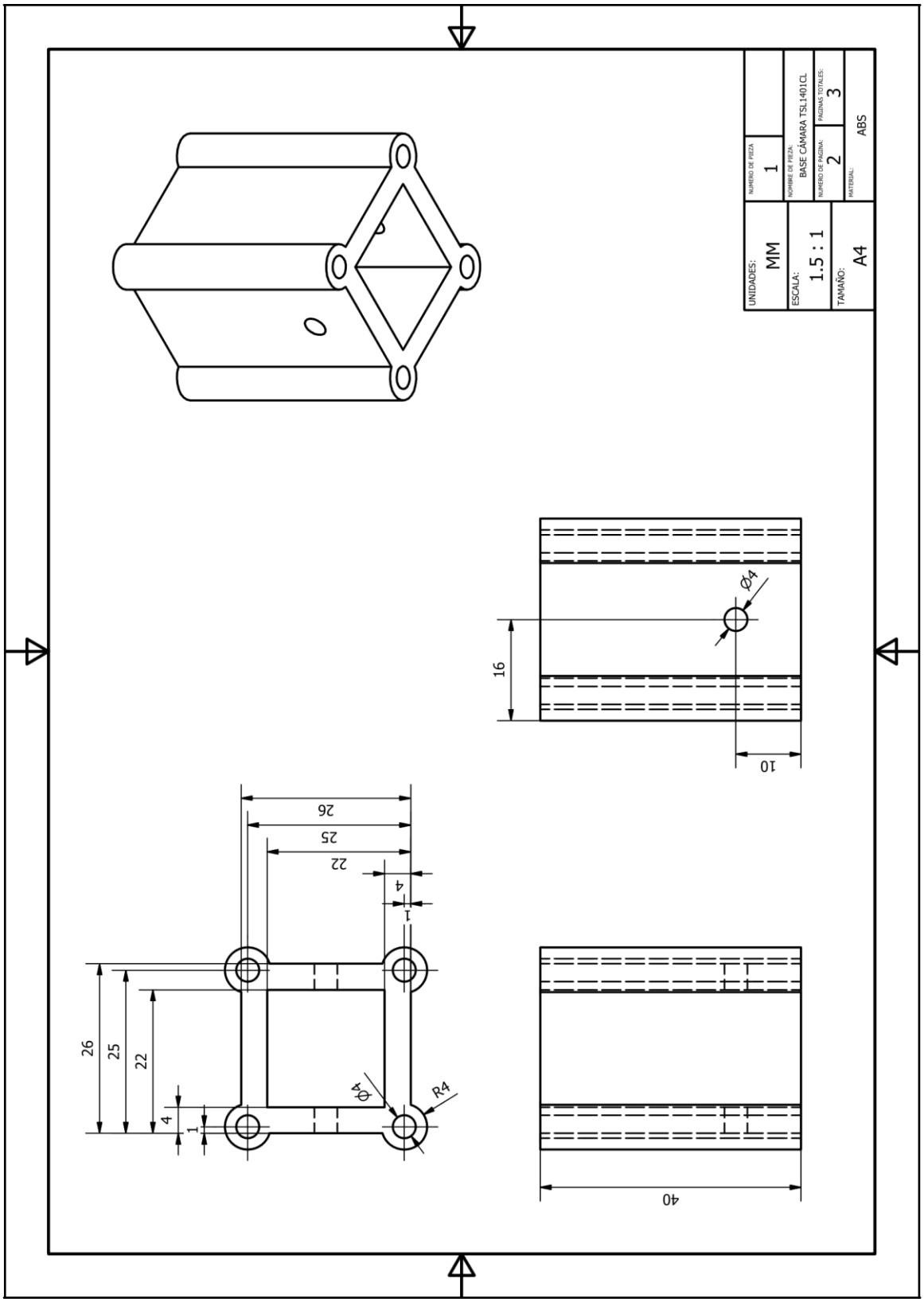

UNIDADES:
MM
ESCALA:
2 : 1
TAMAÑO:
A4
NUMERO DE PIEZA:
6
NOMBRE DE PIEZA:
PIÑÓN CHUCK
NUMERO DE PAGINA:
5
PAGINAS TOTALES:
7
MATERIAL:
ABS
25
4
3
10
R1
R2
R3
Ø10

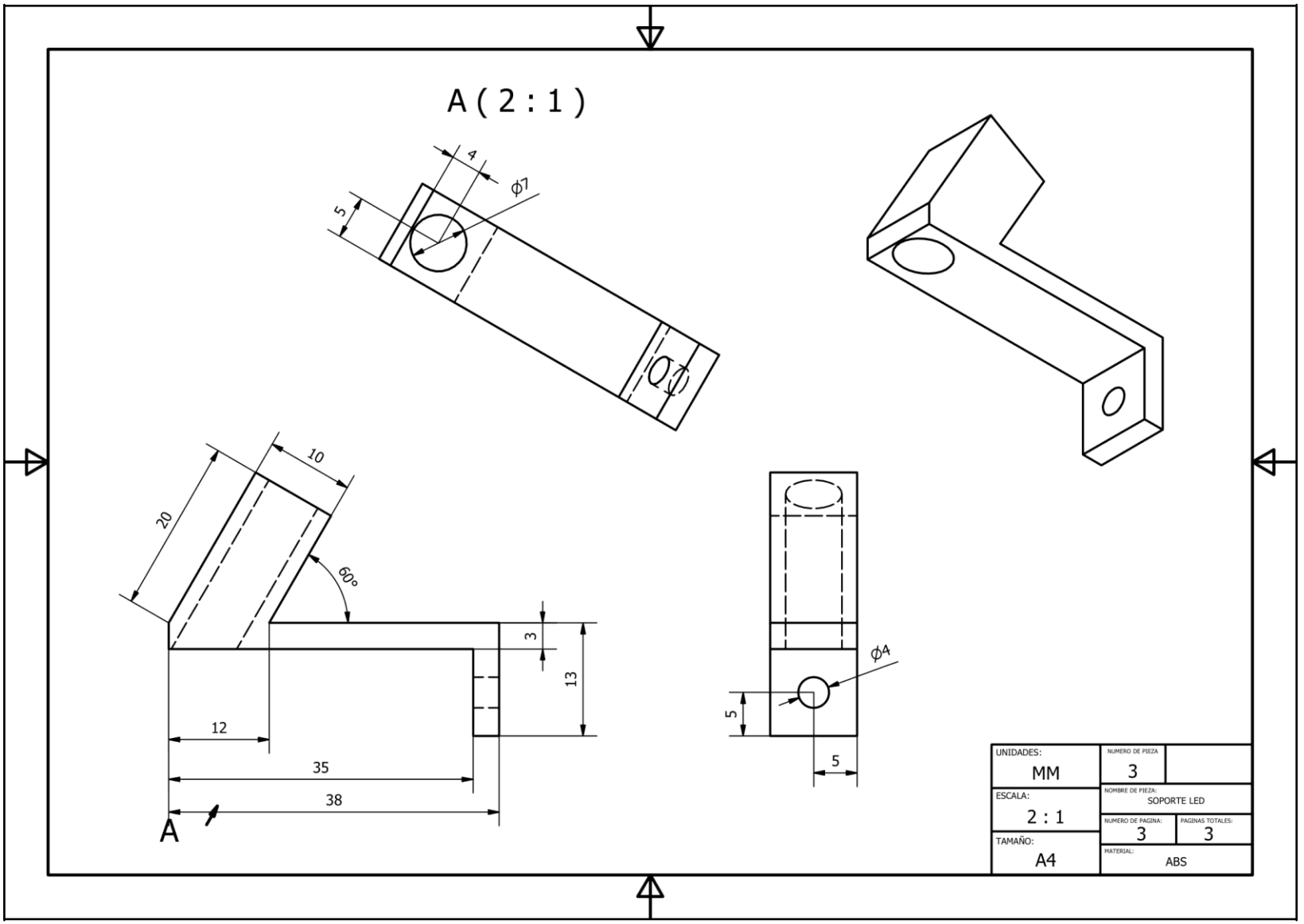

NUMERO DE PIEZA: 8
NOMBRE DE PIEZA: LEVANTA STEPPER
NUMERO DE PAGINA: 6
PAGINAS TOTALES: 7
MATERIAL: ABS
UNIDADES: MM
ESCALA: 1.5 : 1
TAMAÑO: A4
40
R5
4
4
13
23
R15
35
R18
ϕ5
R2
4
12
23
14


UNIDADES:
MM
ESCALA:
3 : 1
TAMAÑO:
A4
NUMERO DE PIEZA
10
NOMBRE DE PIEZA:
TOPE ENDSTOP
NUMERO DE PAGINA:
7
PAGINAS TOTALES:
7
MATERIAL:
MDF
13
5
10
9
Ø3
27
32
37

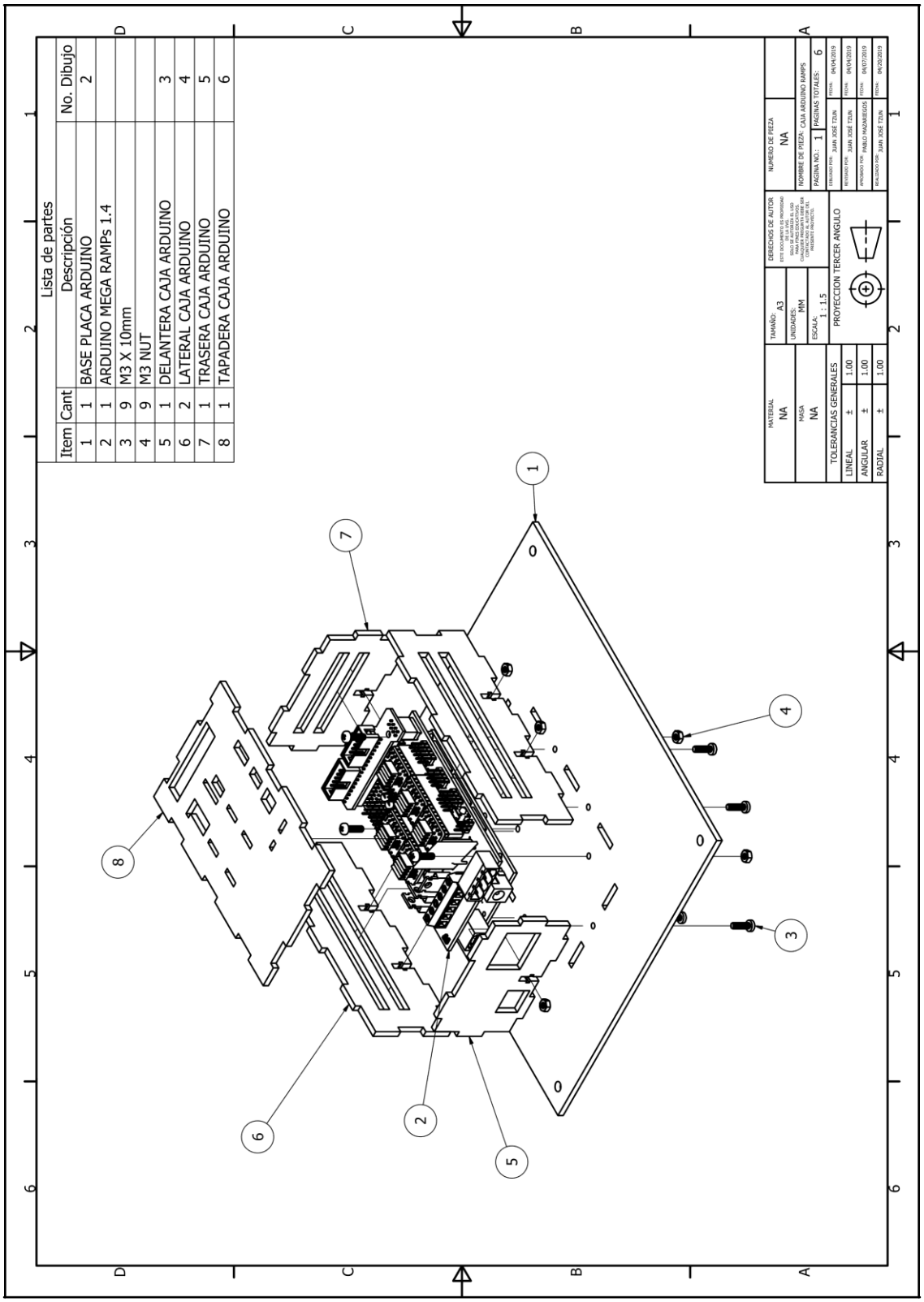
17.9. Planos base cámara linescan

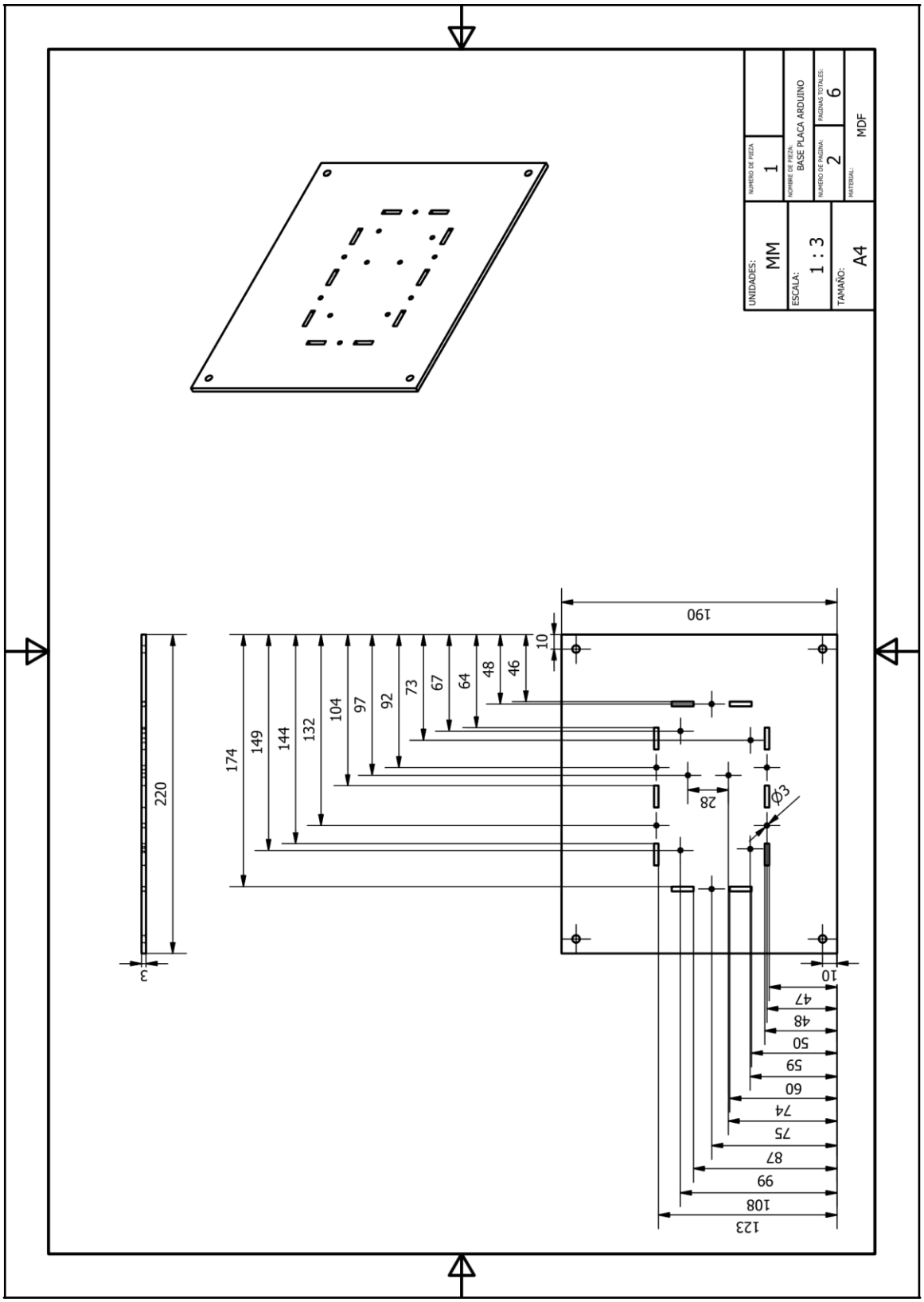


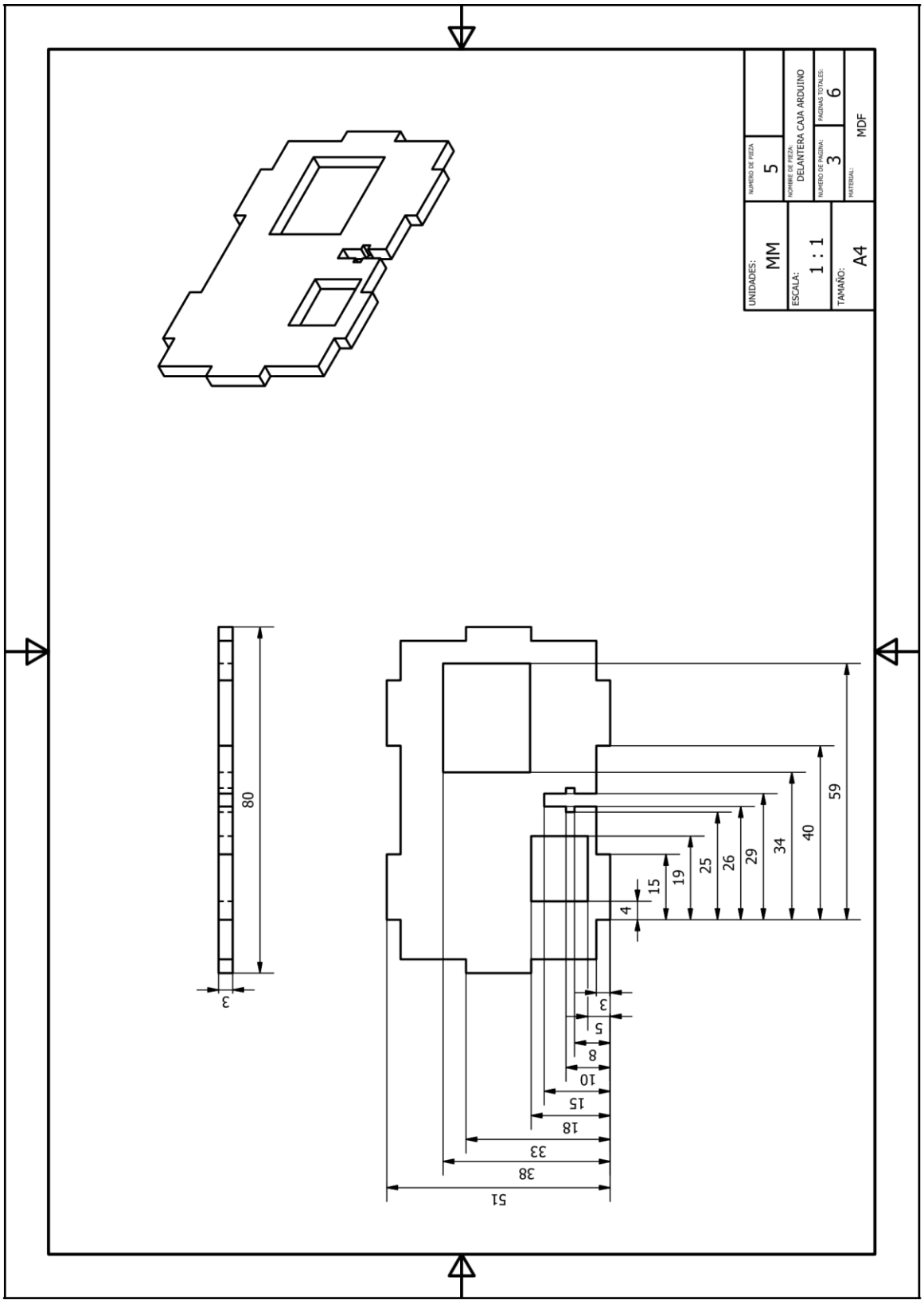

NUMERO DE PIEZA: 1
NOMBRE DE PIEZA: BASE CÁMARA TSL1401CL
NUMERO DE PÁGINA: 2
PÁGINAS TOTALES: 3
MATERIAL: ABS
UNIDADES: MM
ESCALA: 1.5 : 1
TAMAÑO: A4
Ø4
16
10
26
25
22
4
1
Ø4
R4
40

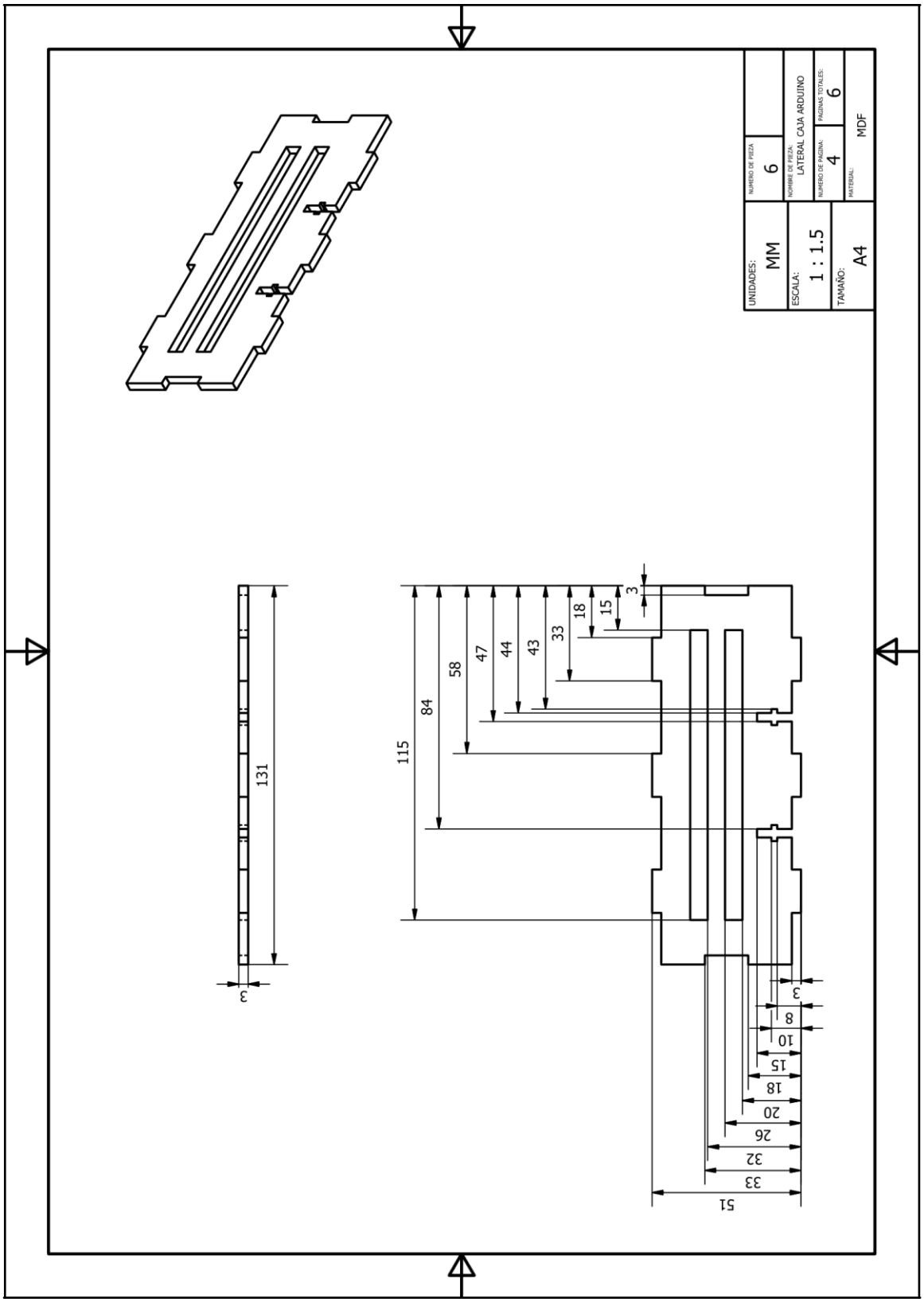

A (2 : 1)
4
ϕ7
5
ϕD
10
20
60°
3
13
12
35
38
A
ϕA
5
5
UNIDADES:
MM
NUMERO DE PIEZA
3
ESCALA:
2 : 1
NOMBRE DE PIEZA:
SOPORTE LED
NUMERO DE PAGINA:
3
PAGINAS TOTALES:
3
TAMAÑO:
A4
MATERIAL:
ABS

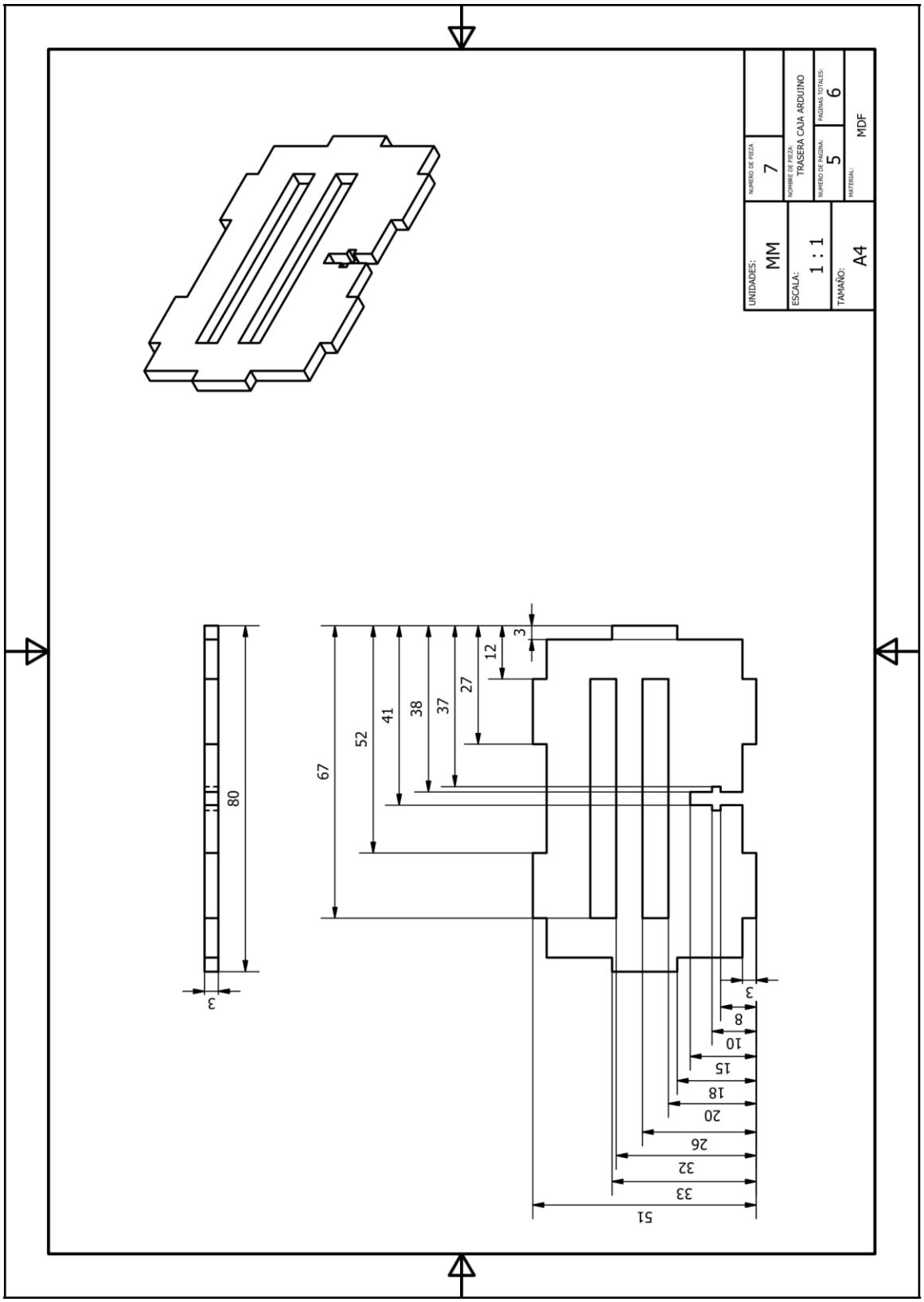
17.10. Planos caja Arduino RAMPs

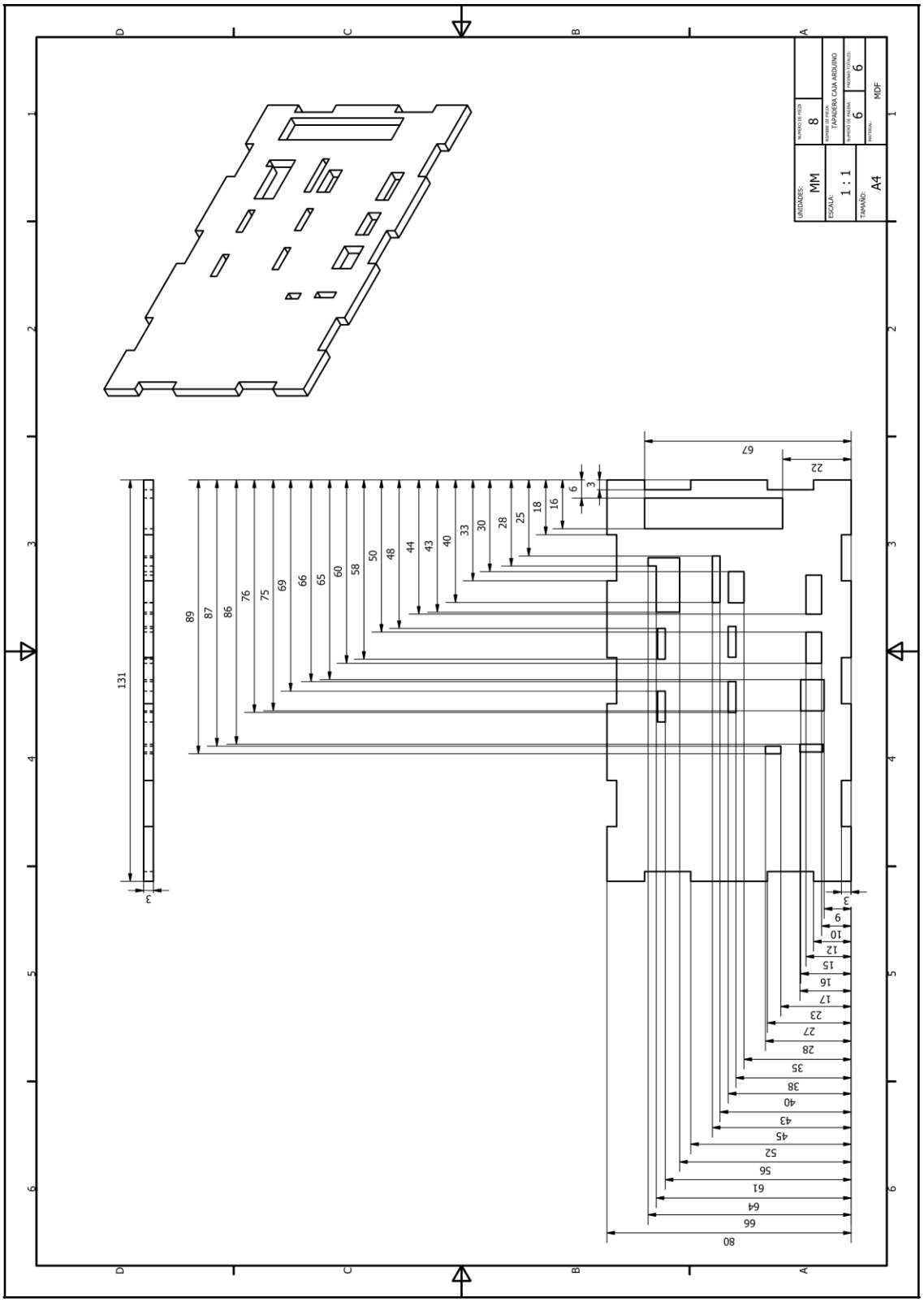



UNIDADES: MM
ESCALA: 1 : 3
TAMAÑO: A4
NUMERO DE PIEZA: 1
NOMBRE DE PIEZA: BASE PLACA ARDUINO
NUMERO DE PAGINA: 2
PAGINAS TOTALES: 6
MATERIAL: MDF
190
10
174
149
144
132
104
97
92
73
67
64
48
46
220
3
28
Φ3
10
47
48
50
59
60
74
75
87
99
108
123

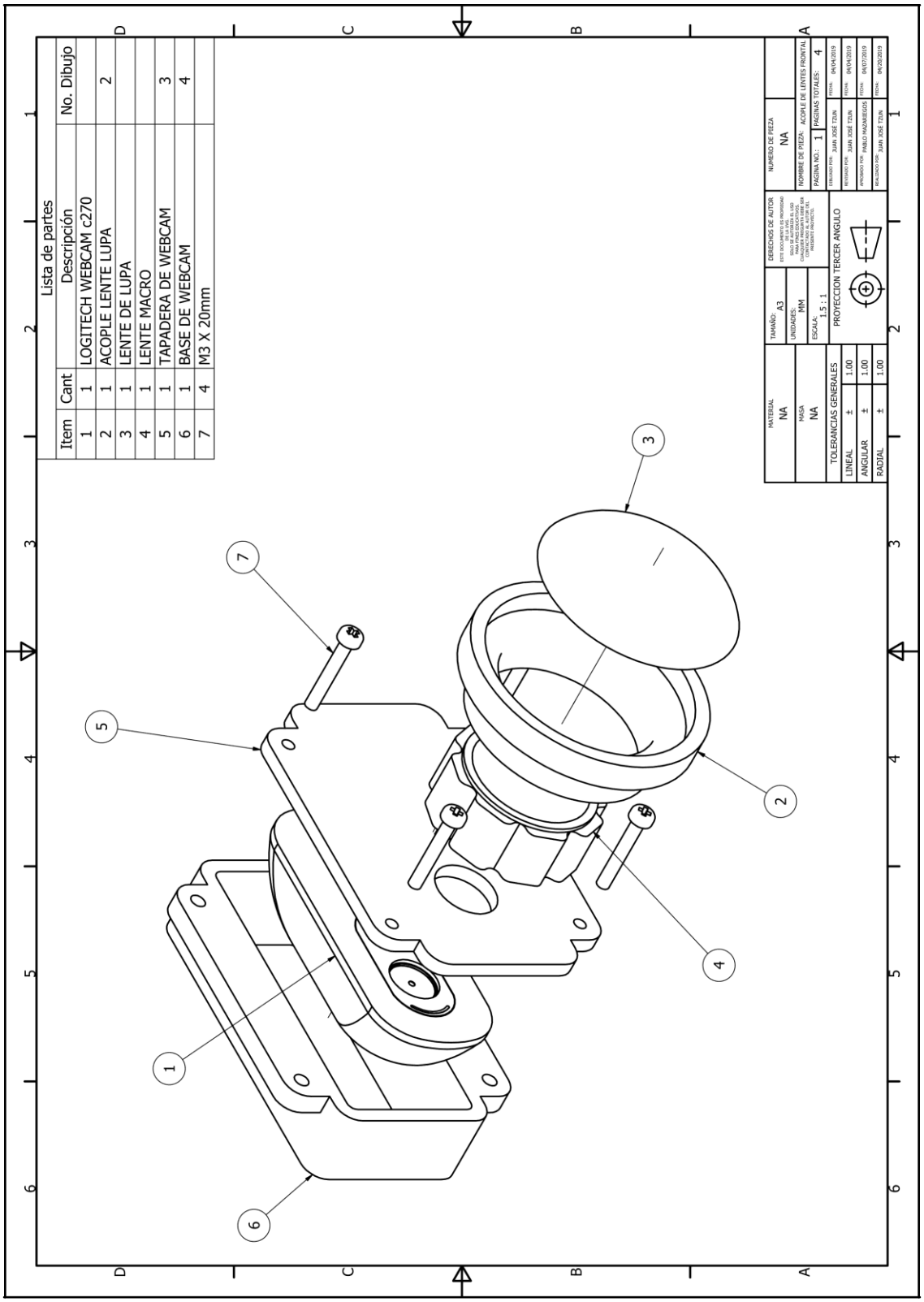

UNIDADES: MM
ESCALA: 1 : 1
TAMAÑO: A4
NUMERO DE PIEZA: 5
NOMBRE DE PIEZA: DELANTERA CAJA ARDUINO
NUMERO DE PAGINA: 3
PAGINAS TOTALES: 6
MATERIAL: MDF
80
3
59
40
34
29
26
25
19
15
4
51
38
33
18
15
10
8
5
3

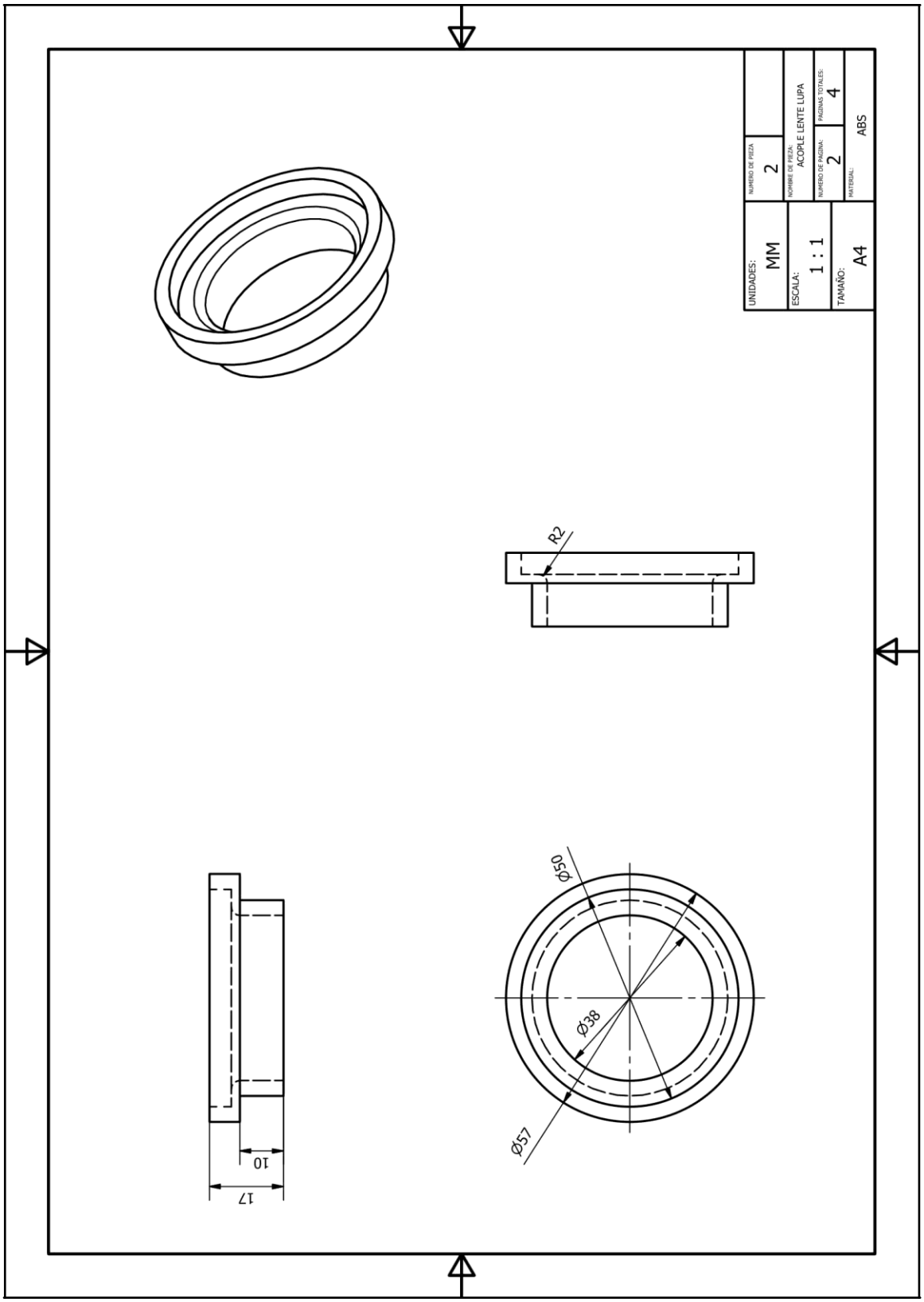

NÚMERO DE PIEZA: 6
NOMBRE DE PIEZA: LATERAL CAJA ARDUINO
NÚMERO DE PÁGINA: 4
PÁGINAS TOTALES: 6
MATERIAL: MDF
UNIDADES: MM
ESCALA: 1 : 1.5
TAMAÑO: A4
131
3
115
84
58
47
44
43
33
18
15
3
51
33
32
26
20
18
15
10
8
3

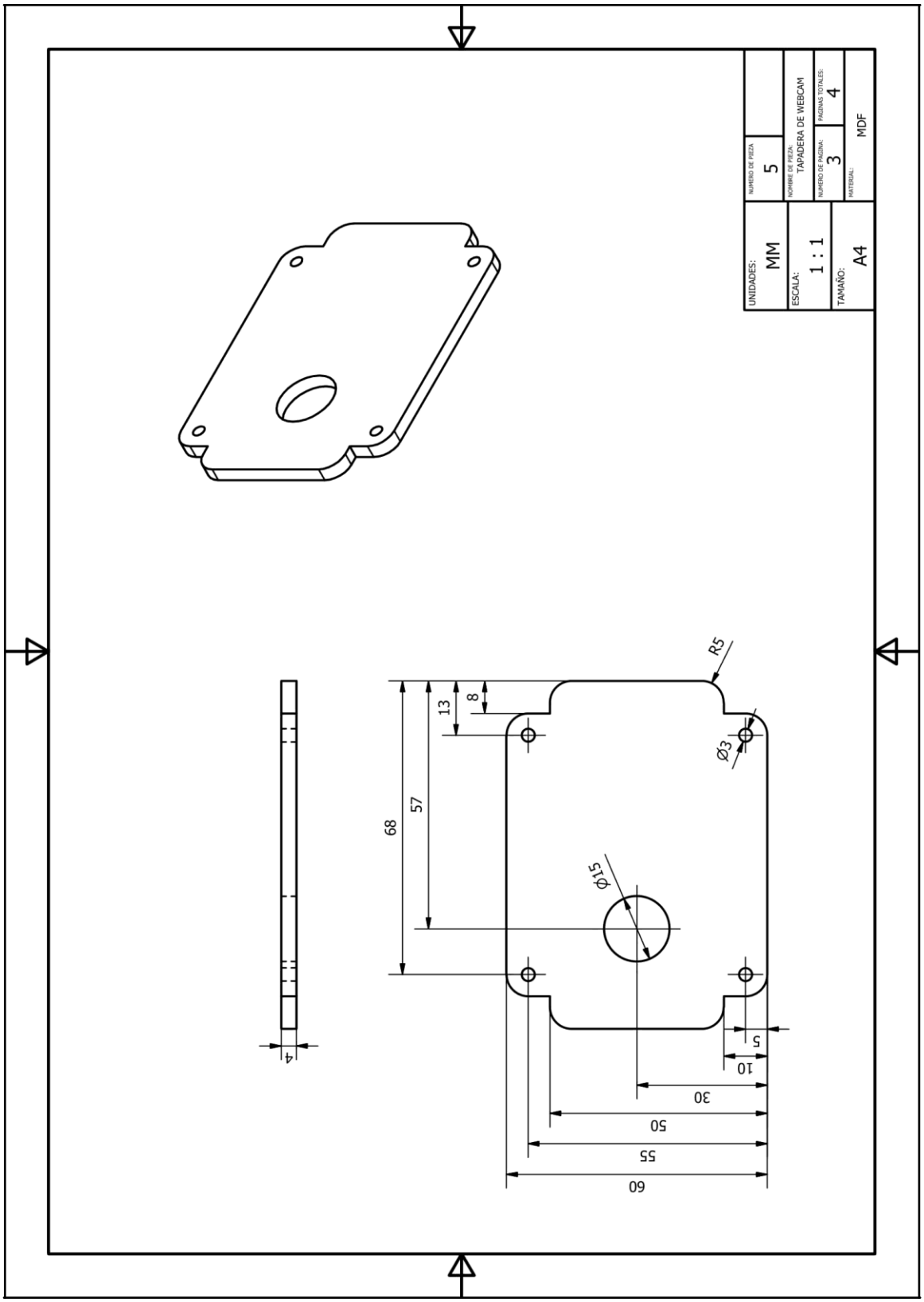

NUMERO DE PIEZA: 7
NOMBRE DE PIEZA: TRASERA CAJA ARDUINO
NUMERO DE PAGINA: 5
PAGINAS TOTALES: 6
MATERIAL: MDF
UNIDADES: MM
ESCALA: 1 : 1
TAMAÑO: A4
80
3
67
52
41
38
37
27
12
3
3
8
10
15
18
20
26
32
33
51

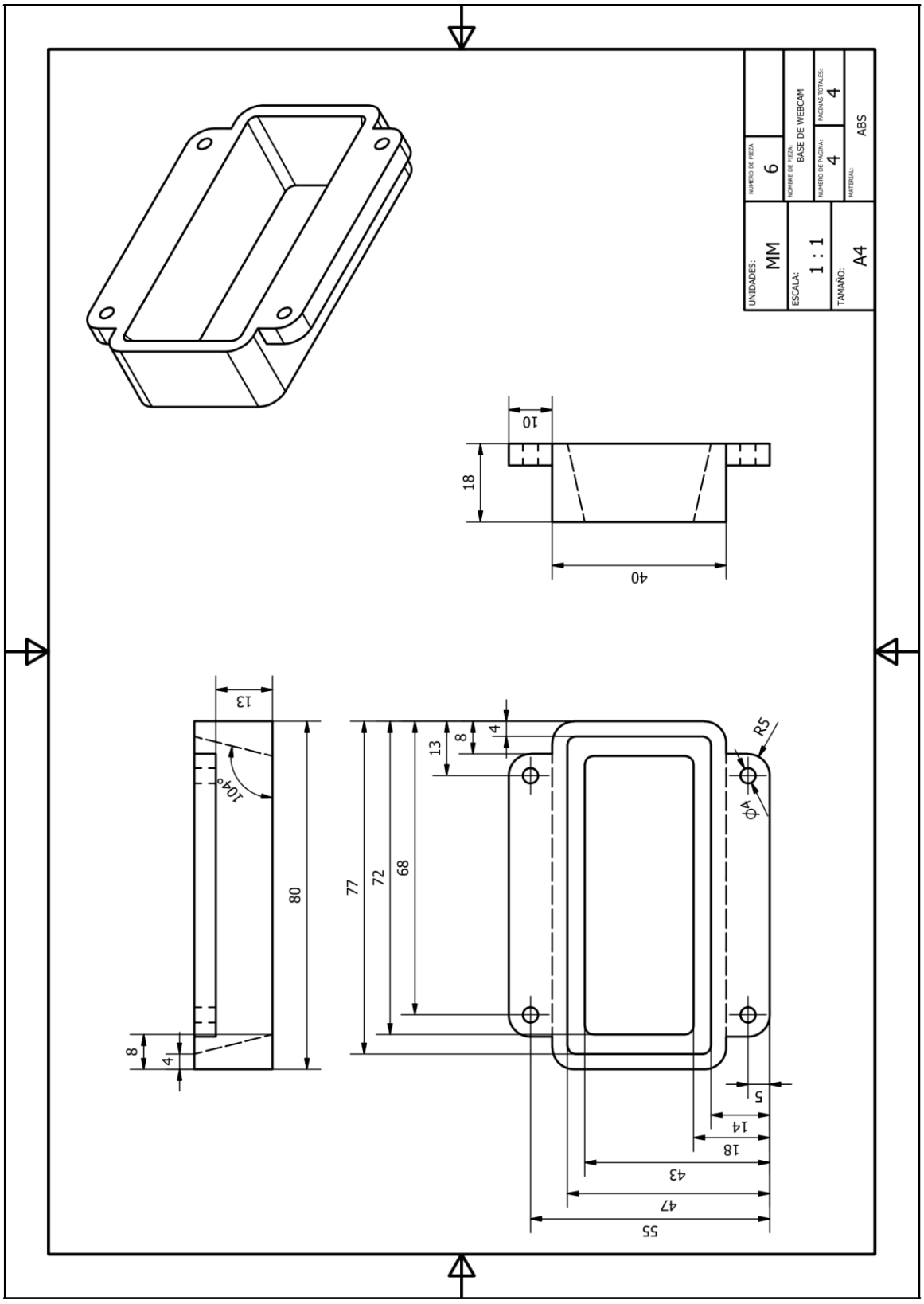

NÚMERO DE PIEZA:
8
NOMBRE DE PIEZA:
TAPADERA CAJA ARDUINO
NÚMERO DE PLANO:
6
PLANOS TOTALES:
6
UNIDADES:
MM
ESCALA:
1 : 1
TAMAÑO:
A4
MATERIAL:
MDF
67
22
3
6
18
16
25
28
30
33
40
43
44
48
50
58
60
65
66
69
75
76
86
87
89
131
3
3
9
10
12
15
16
17
23
27
28
35
38
40
43
45
52
56
61
64
66
80

17.11. Planos acople de lentes frontal

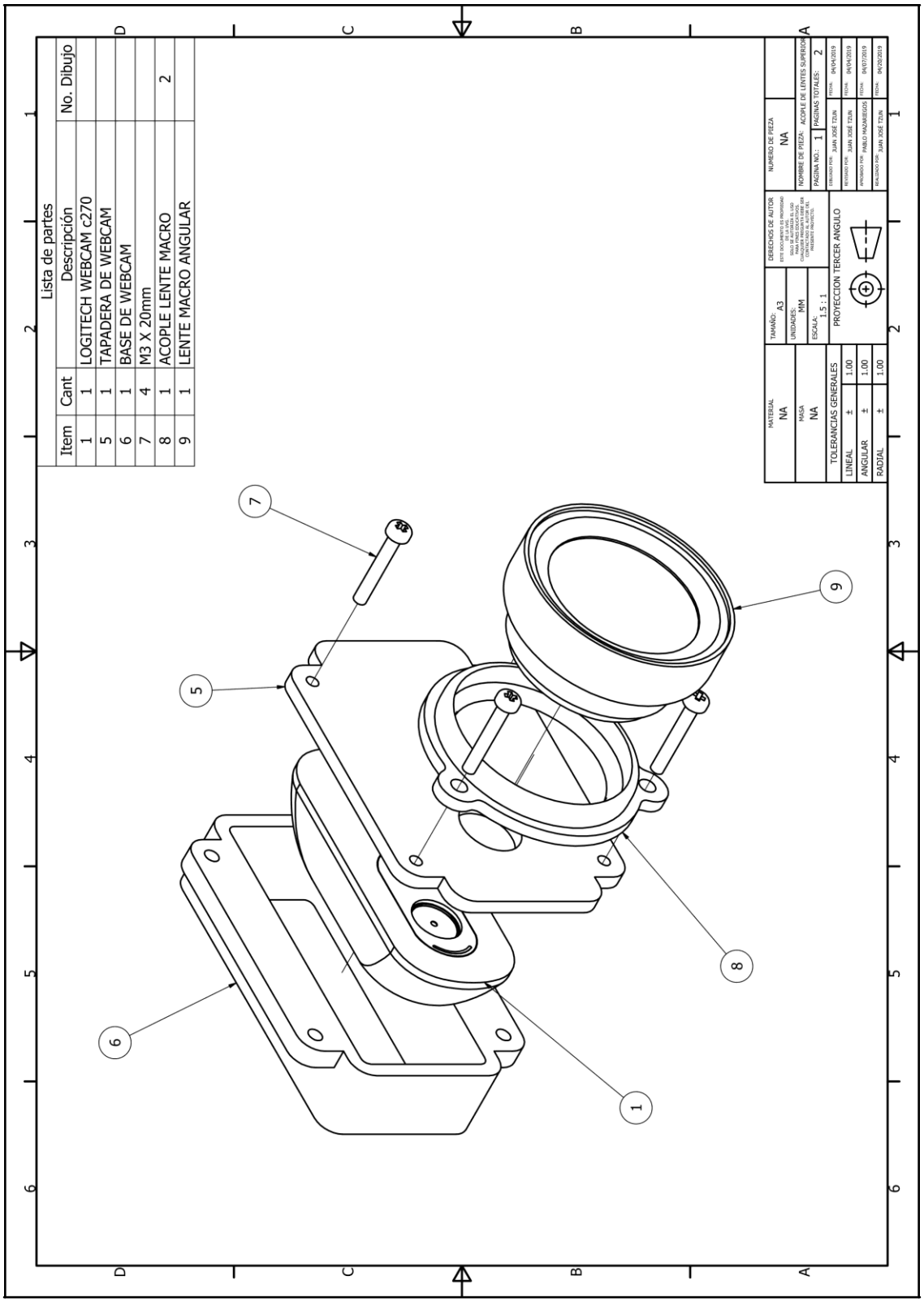


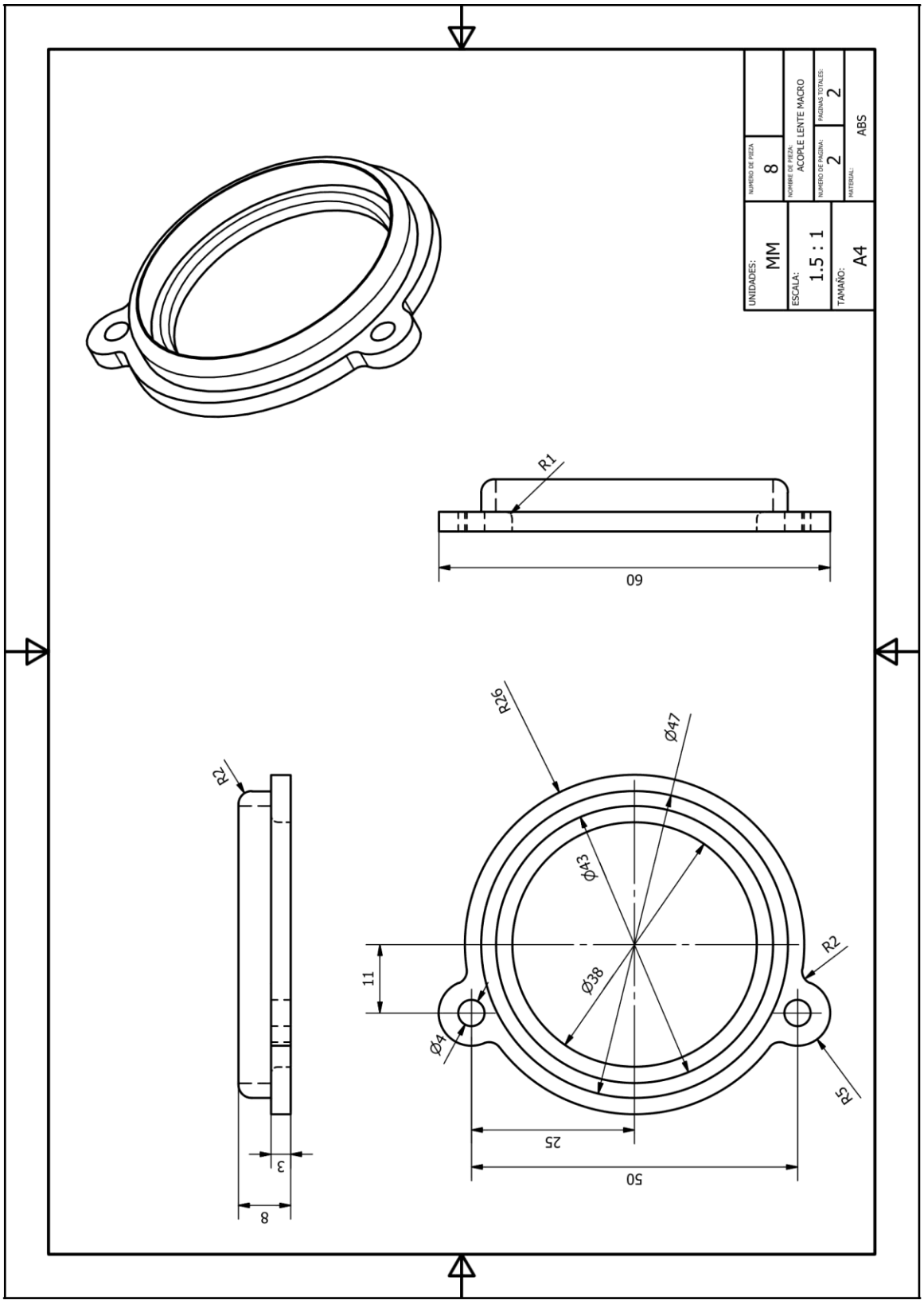

NUMERO DE PIEZA: 2
NOMBRE DE PIEZA: ACOPLE LENTE LUPA
NUMERO DE PAGINA: 2
PAGINAS TOTALES: 4
MATERIAL: ABS
UNIDADES: MM
ESCALA: 1 : 1
TAMAÑO: A4
R2
Ø50
Ø38
Ø57
10
17


UNIDADES: MM
ESCALA: 1 : 1
TAMAÑO: A4
NUMERO DE PIEZA: 5
NOMBRE DE PIEZA: TAPADERA DE WEBCAM
NUMERO DE PAGINA: 3
PAGINAS TOTALES: 4
MATERIAL: MDF
R5
Ø3
Ø15
13
8
68
57
4
5
10
30
50
55
60

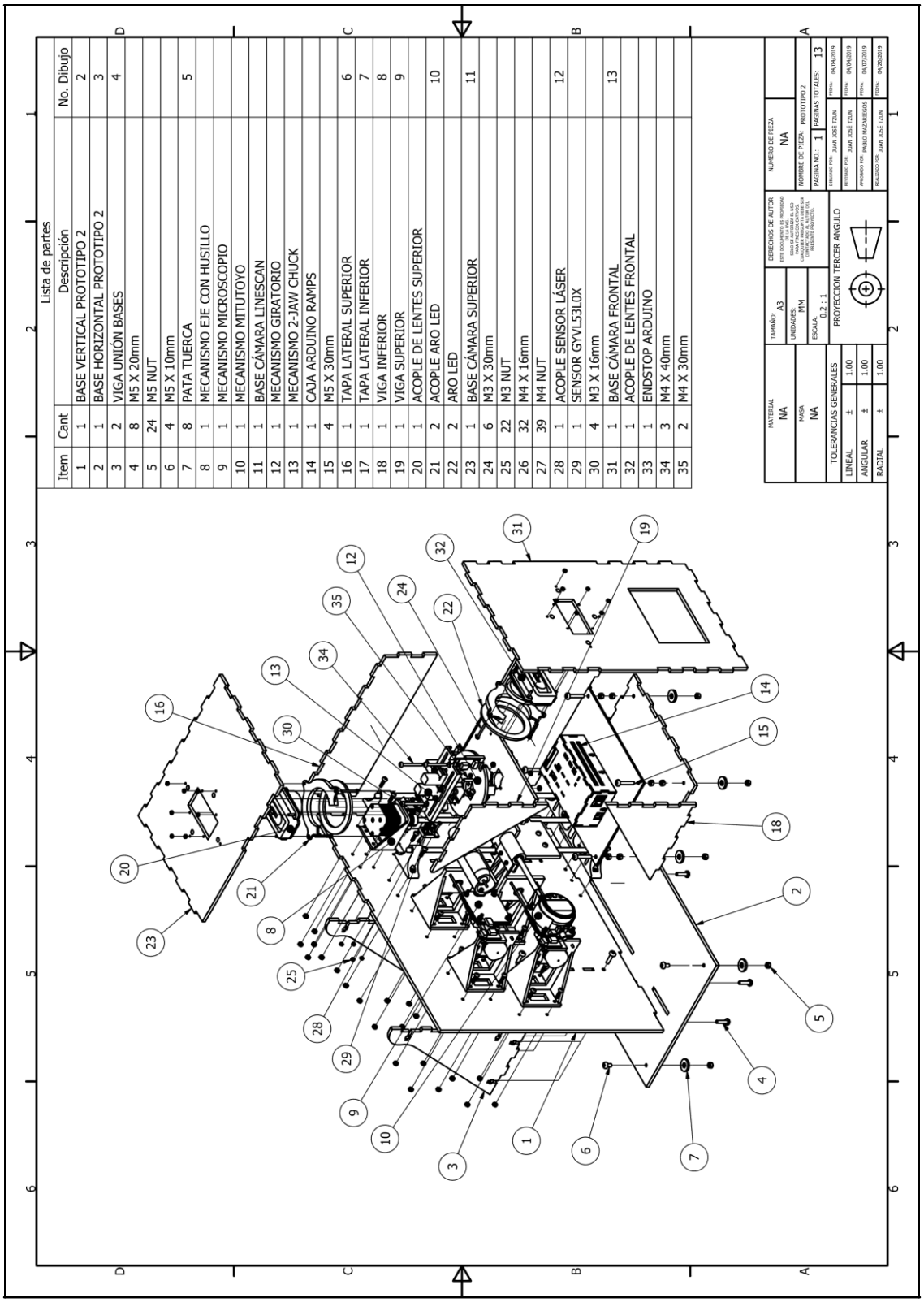

NÚMERO DE PIEZA: 6
NOMBRE DE PIEZA: BASE DE WEBCAM
NÚMERO DE PÁGINA: 4
PÁGINAS TOTALES: 4
MATERIAL: ABS
UNIDADES: MM
ESCALA: 1 : 1
TAMAÑO: A4
10
18
40
13
104°
80
8
4
77
72
68
13
8
4
R5
ϕA
5
14
18
43
47
55

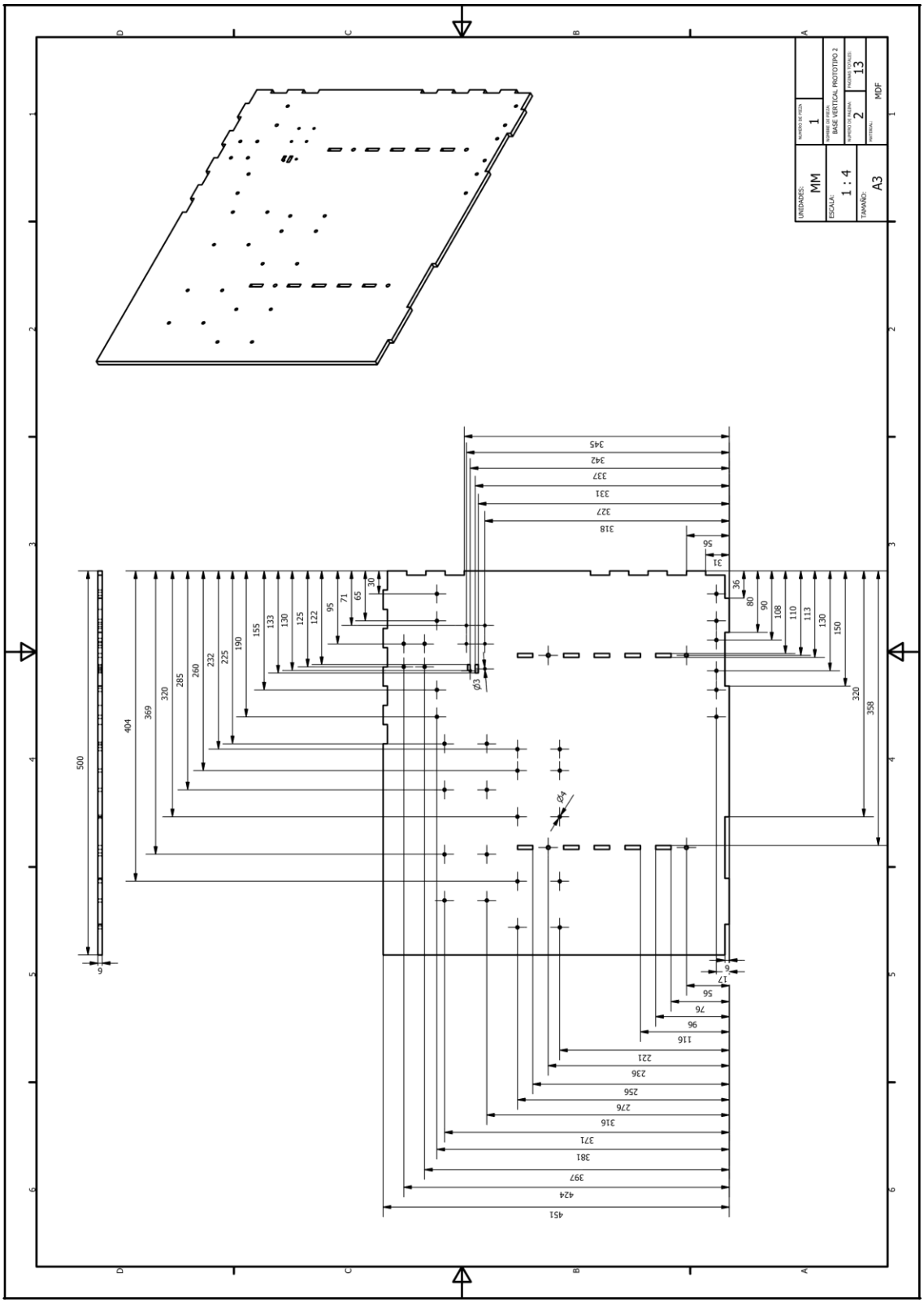
17.12. Planos acople de lentes superior

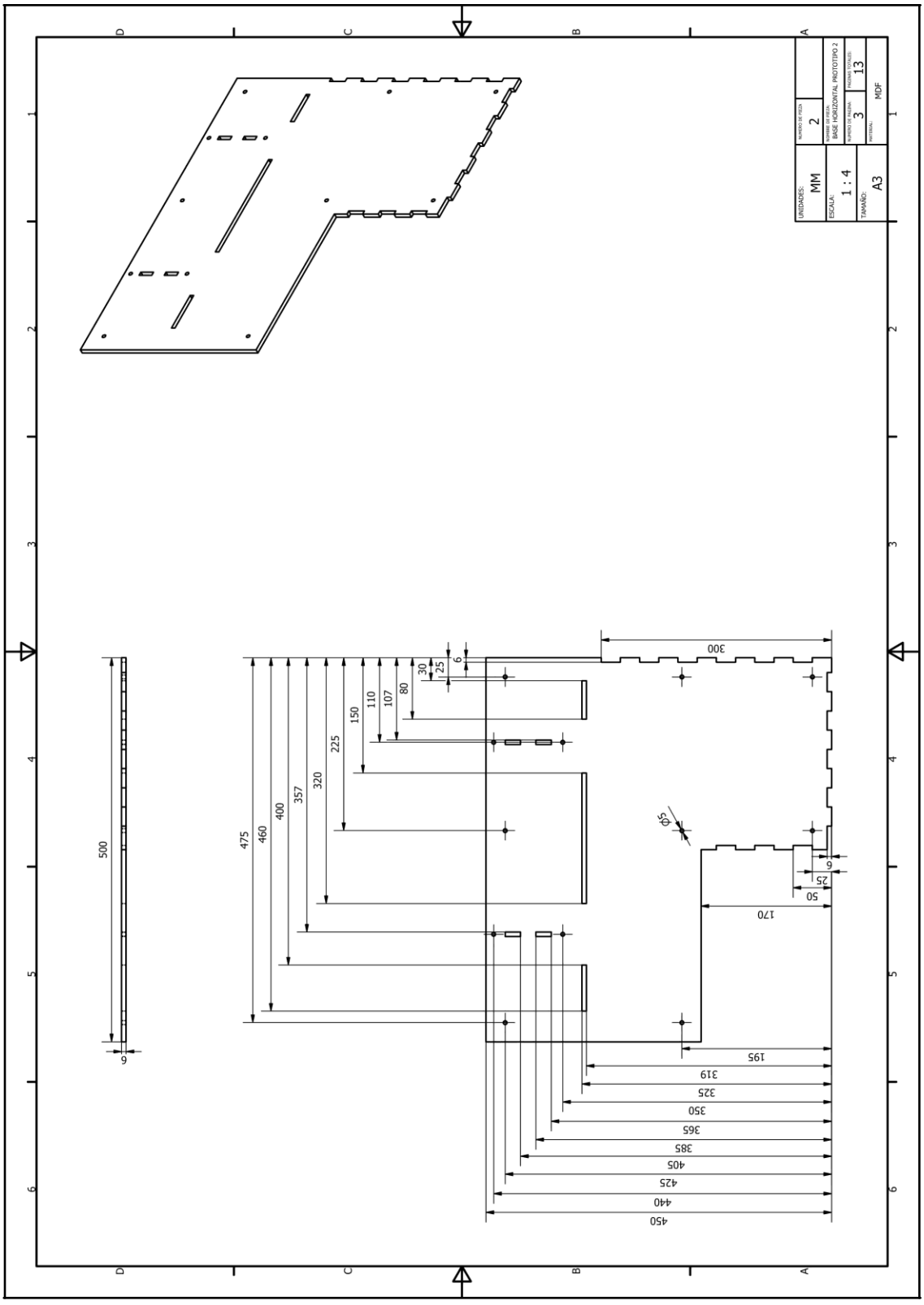


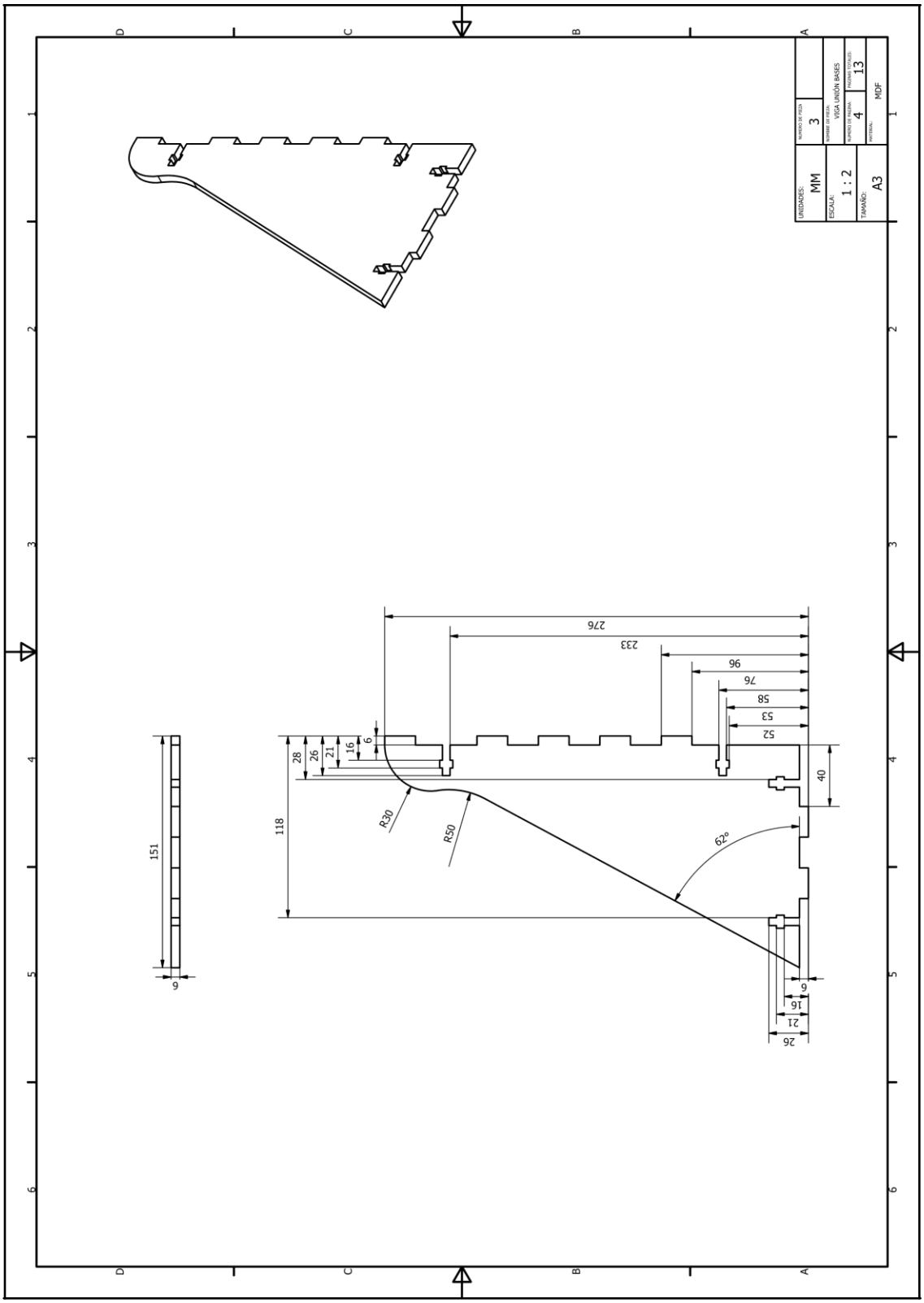

NÚMERO DE PIEZA: 8
NOMBRE DE PIEZA: ACOPLE LENTE MACRO
NÚMERO DE PÁGINA: 2
PÁGINAS TOTALES: 2
MATERIAL: ABS
UNIDADES: MM
ESCALA: 1.5 : 1
TAMAÑO: A4
R1
60
R26
Ø47
Ø43
R2
11
Ø38
Ø4
R5
25
50
R2
3
8

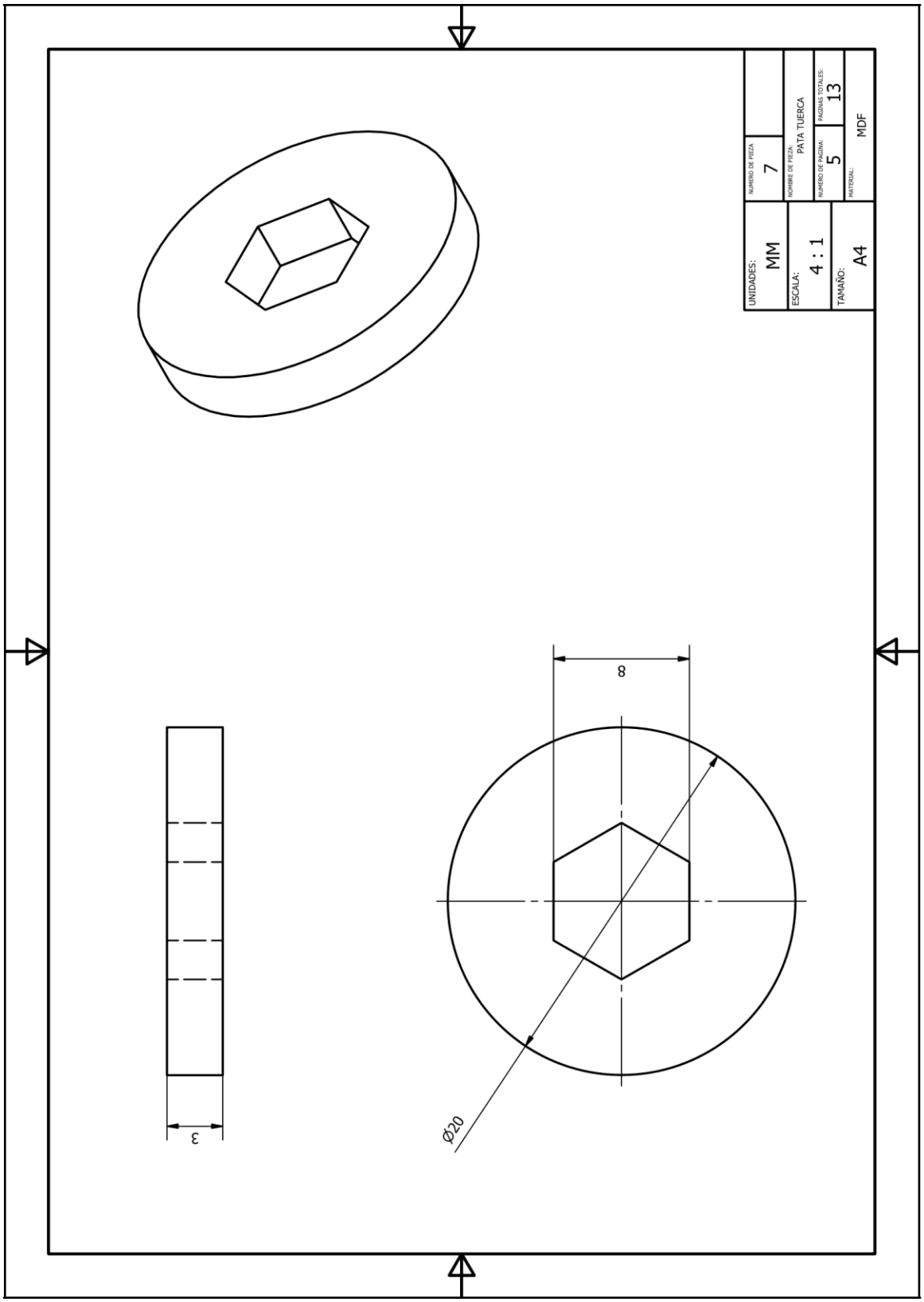
17.13. Planos prototipo 2

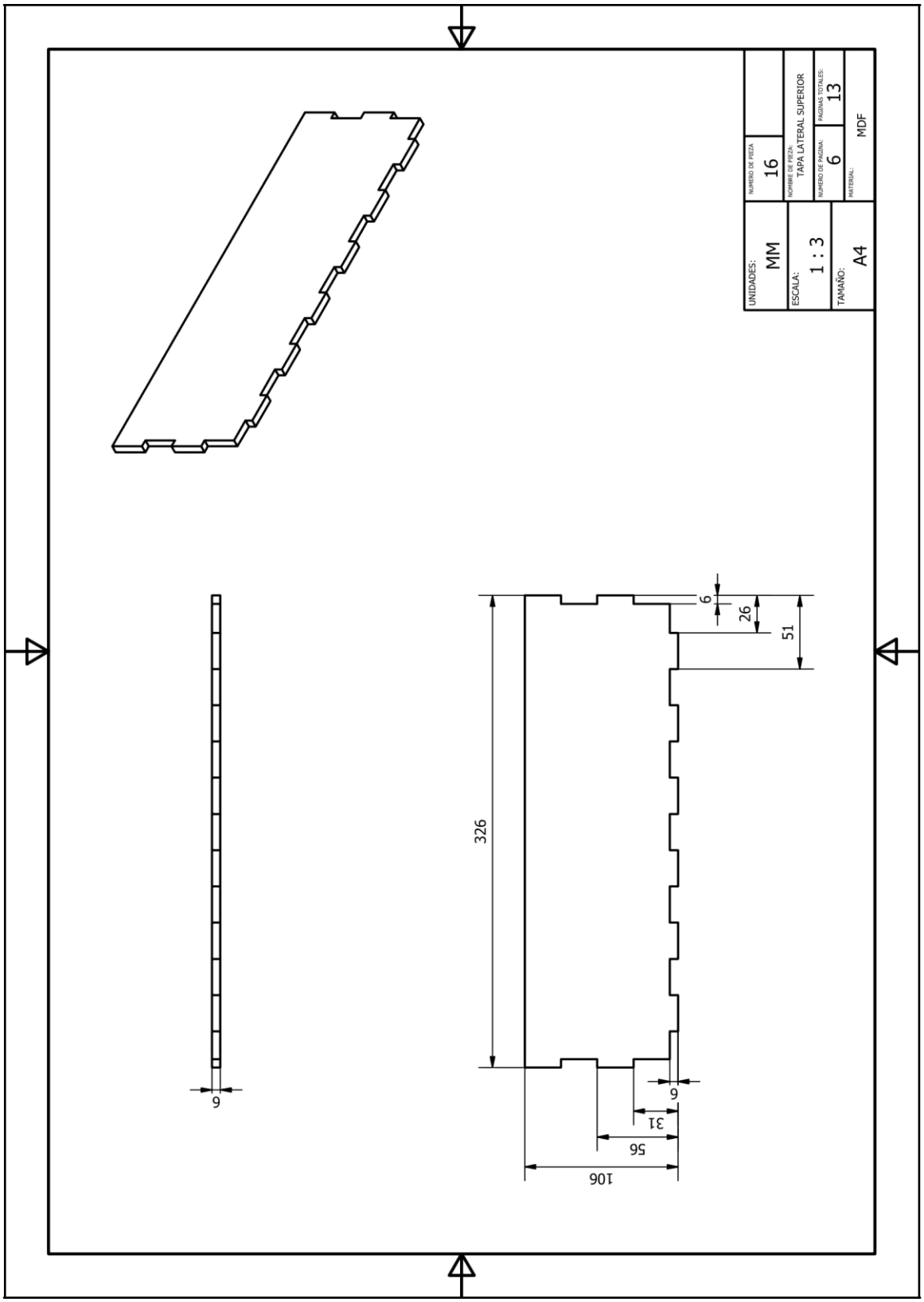


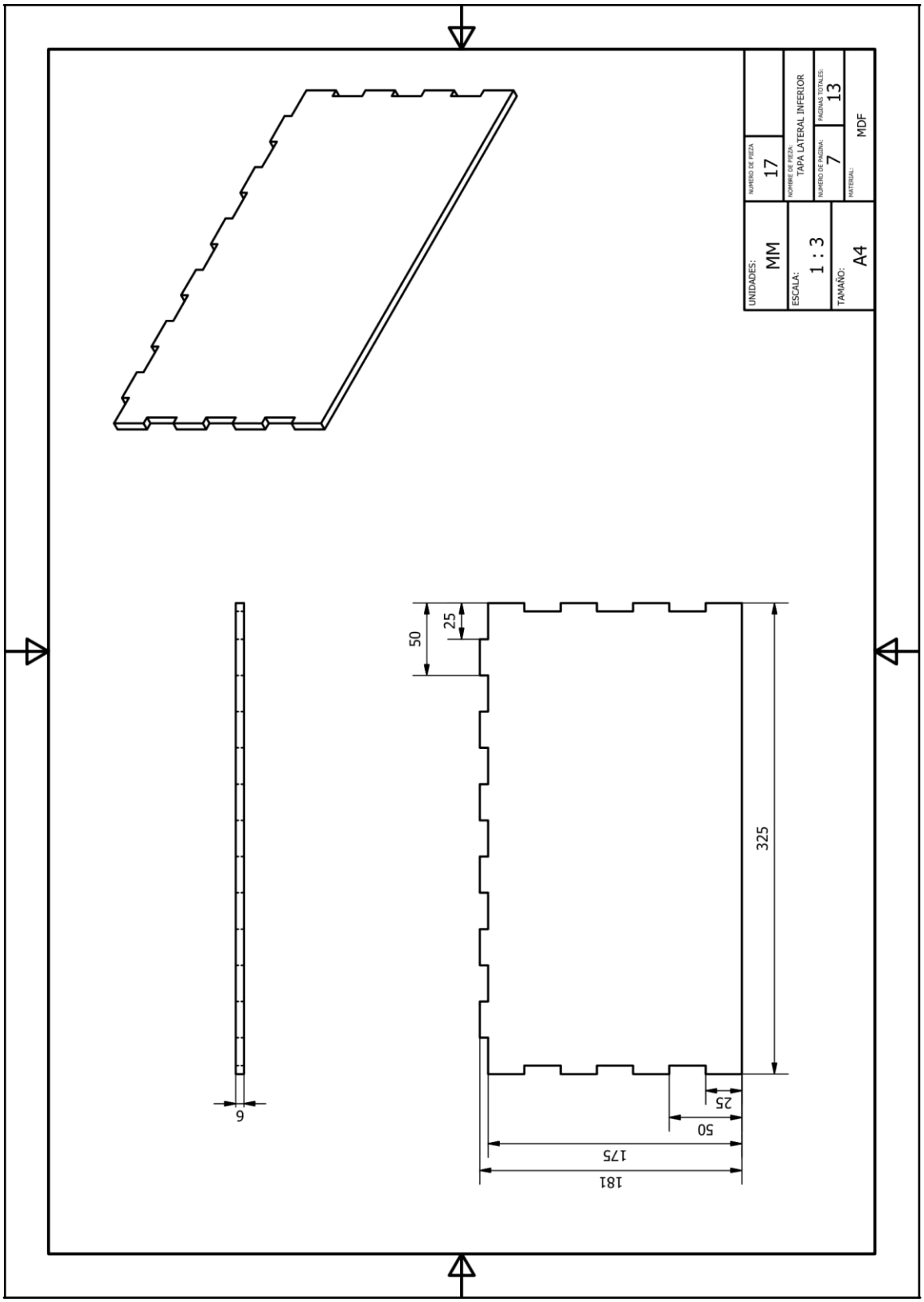

NÚMERO DE PEZA
1
NOMBRE DE PIEZA:
BASE VERTICAL PROTOTIPO 2
NÚMERO DE PIEZAS:
2
PIEZAS TOTALES:
13
MATERIAL:
MDF
UNIDADES:
MM
ESCALA:
1 : 4
TAMAÑO:
A3
345
342
337
331
327
318
56
31
26
80
90
108
110
113
130
150
320
358
30
71
65
95
122
125
130
133
155
190
225
232
260
285
320
369
404
500
Ø3
Ø4
9
6
17
56
76
96
116
221
236
256
276
316
371
381
397
424
451

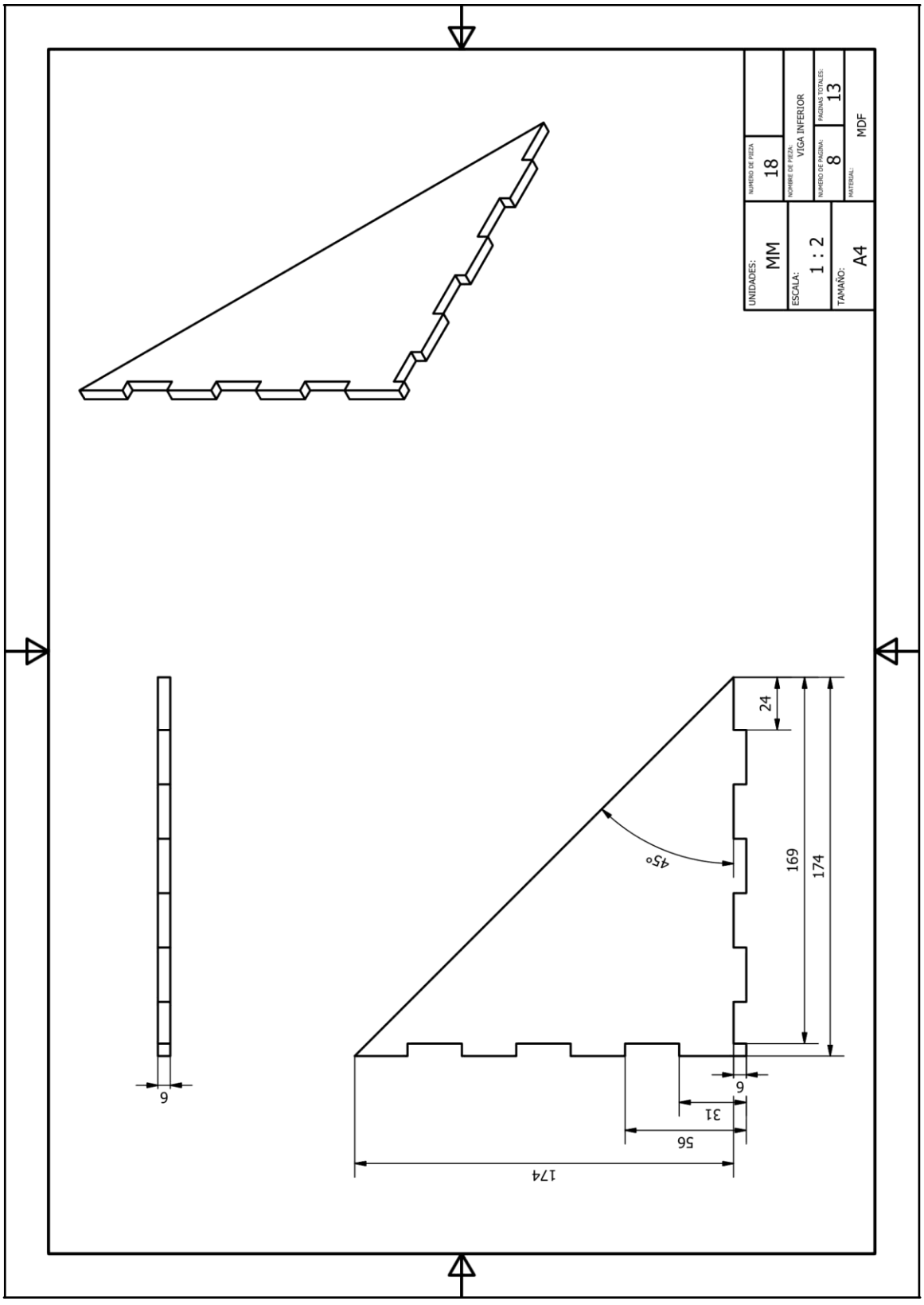

UNIDADES: MM
ESCALA: 1 : 4
TAMAÑO: A3
NÚMERO DE PIEZA 2
NOMBRE DE PIEZA BASE HORIZONTAL PROTOTIPO 2
NÚMERO DE PLIEGO 3
HOJAS TOTALES 13
MATERIAL MDF
500
9
475
460
400
357
320
225
150
110
107
80
30
25
6
300
ϕ5
6
25
50
170
195
319
325
350
365
385
405
425
440
450

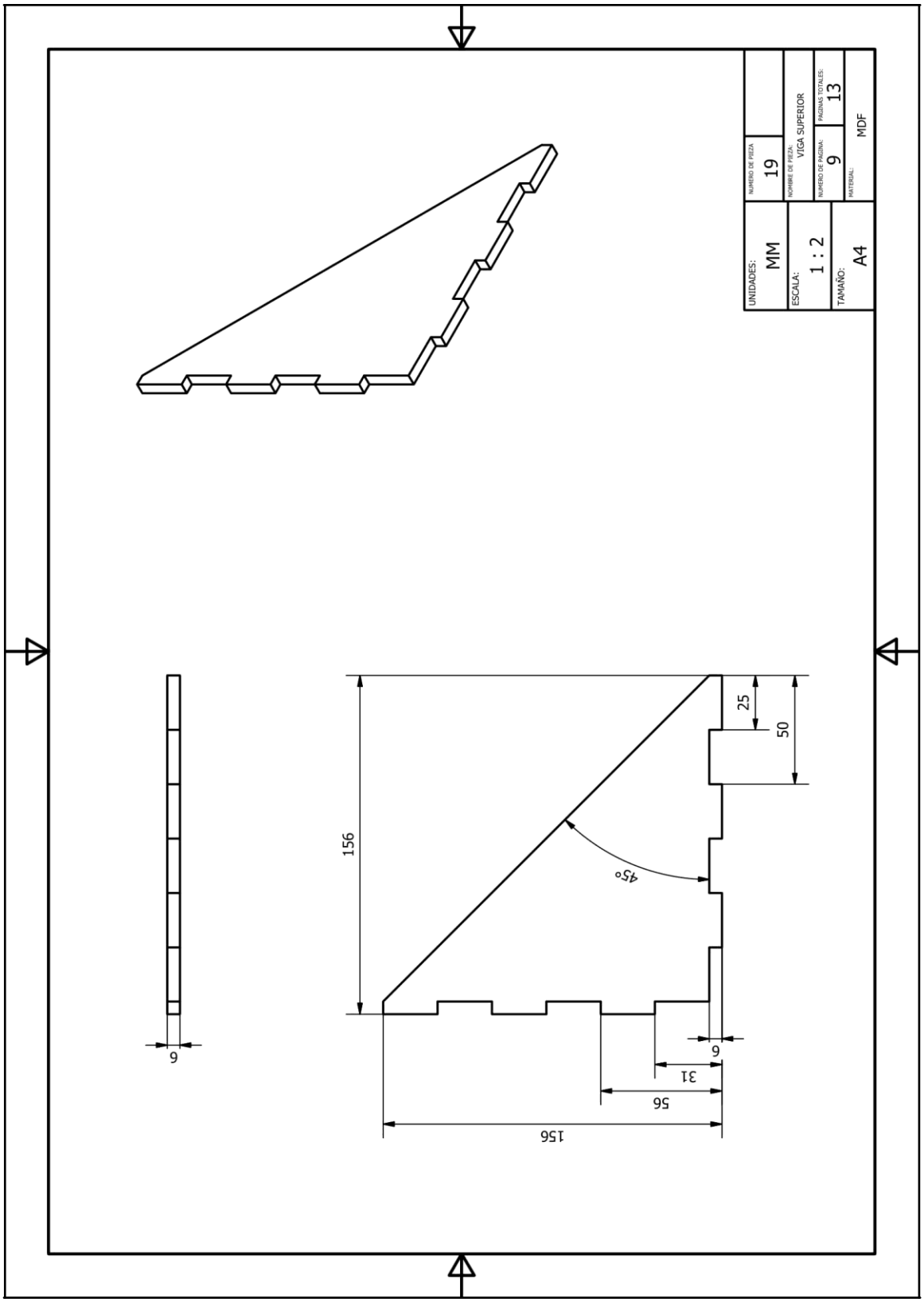

UNIDADES:
MM
ESCALA:
1 : 2
TAMAÑO:
A3
NÚMERO DE HOJA
3
NÚMERO DE HOJAS
4
NOMBRE DE PIEZA:
VIGA UNIÓN BASES
NÚMERO DE PIEZA:
13
MATERIAL:
MDF
276
233
96
76
58
53
52
40
28
26
21
16
6
118
R30
R50
62°
151
6
6
16
21
26

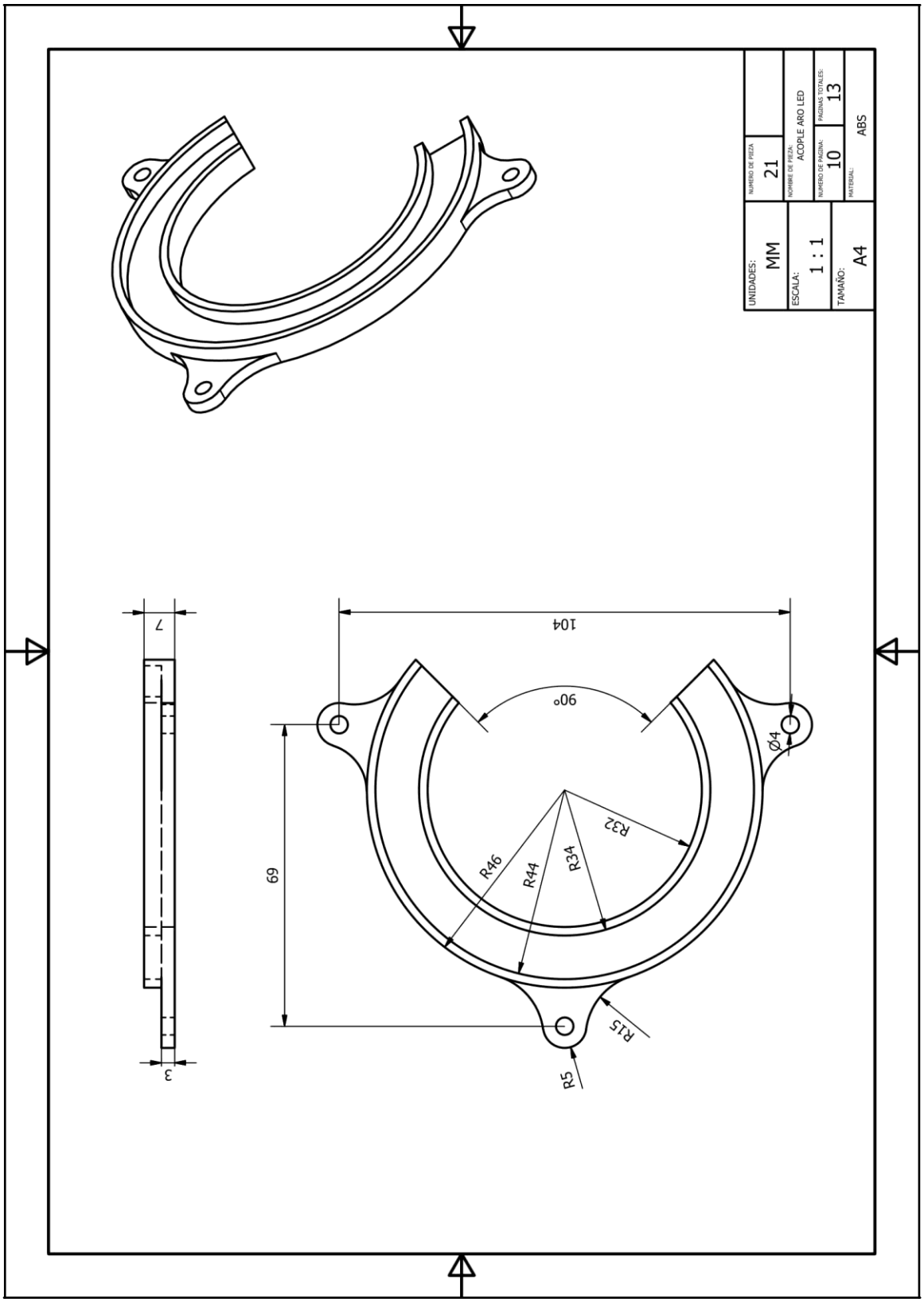

UNIDADES:
MM
ESCALA:
4 : 1
TAMAÑO:
A4
NUMERO DE PIEZA:
7
NOMBRE DE PIEZA:
PATA TUERCA
NUMERO DE PAGINA:
5
PAGINAS TOTALES:
13
MATERIAL:
MDF
8
3
Ø20

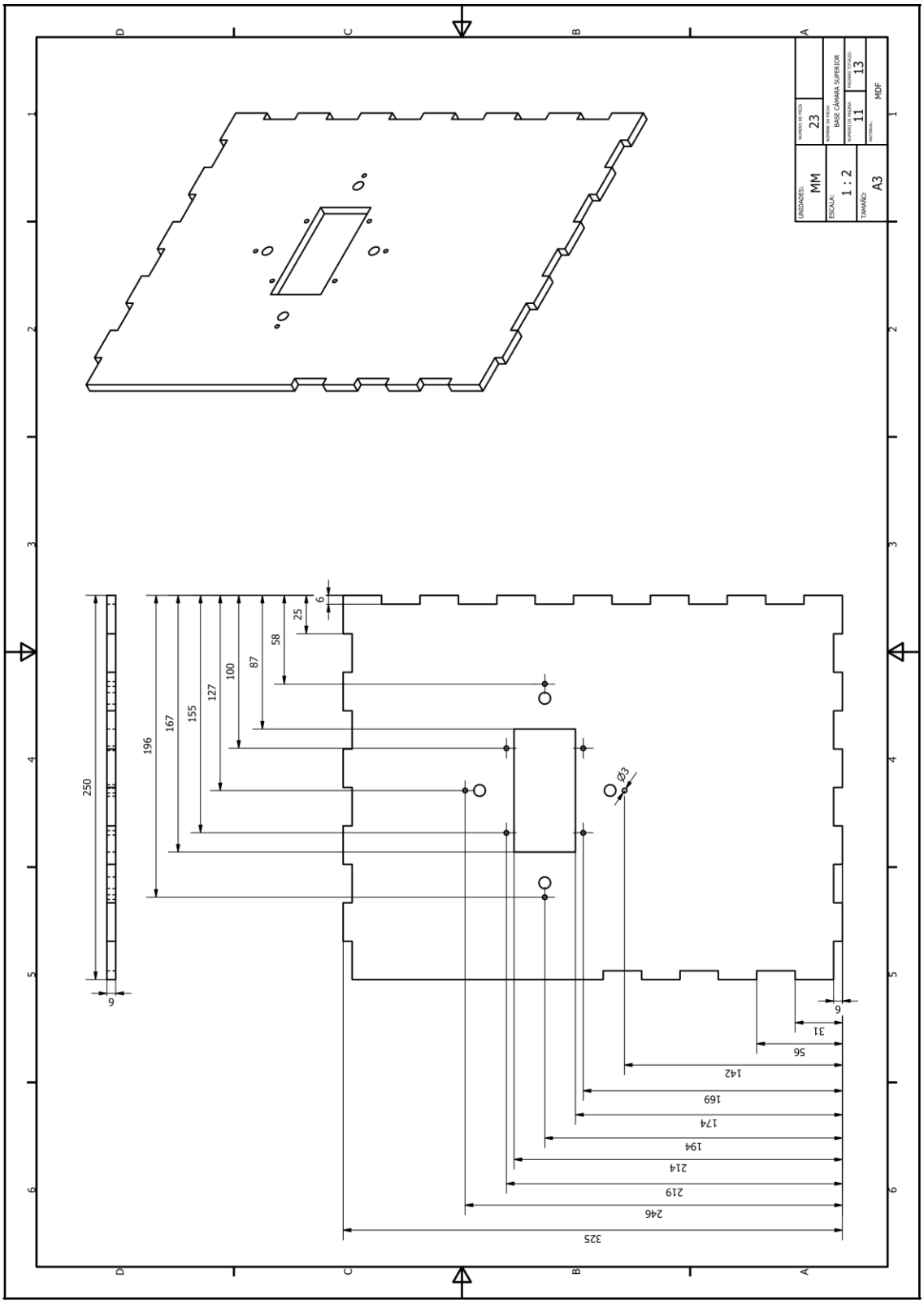

NÚMERO DE PIEZA:
16
NOMBRE DE PIEZA:
TAPA LATERAL SUPERIOR
NÚMERO DE PÁGINA:
6
PÁGINAS TOTALES:
13
UNIDADES:
MM
ESCALA:
1 : 3
TAMAÑO:
A4
MATERIAL:
MDF
6
326
6
26
51
6
31
56
106

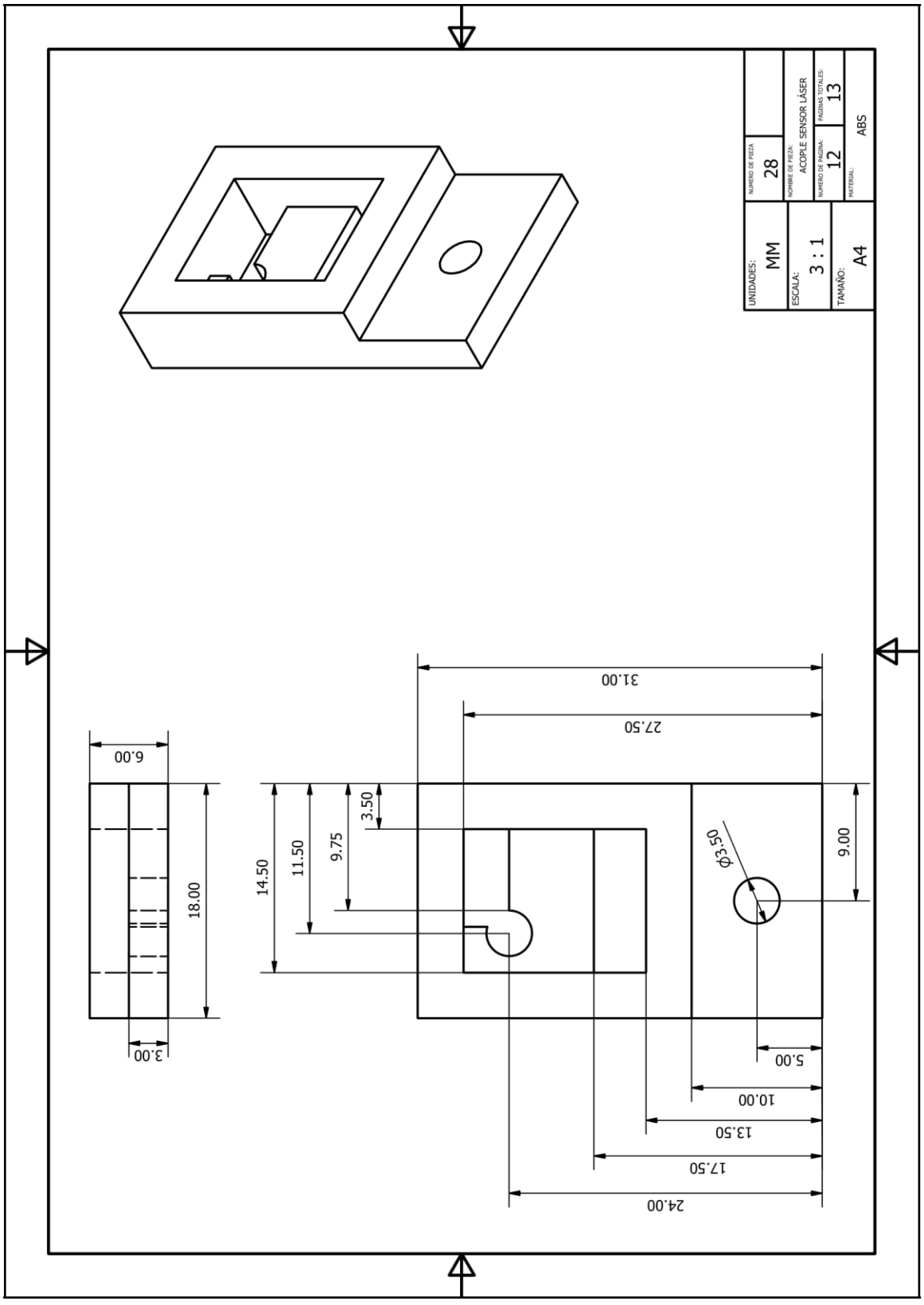

NUMERO DE PIEZA:
17
NOMBRE DE PIEZA:
TAPA LATERAL INFERIOR
NUMERO DE PAGINA:
7
PAGINAS TOTALES:
13
MATERIAL:
MDF
UNIDADES:
MM
ESCALA:
1 : 3
TAMAÑO:
A4
50
25
325
6
25
50
175
181

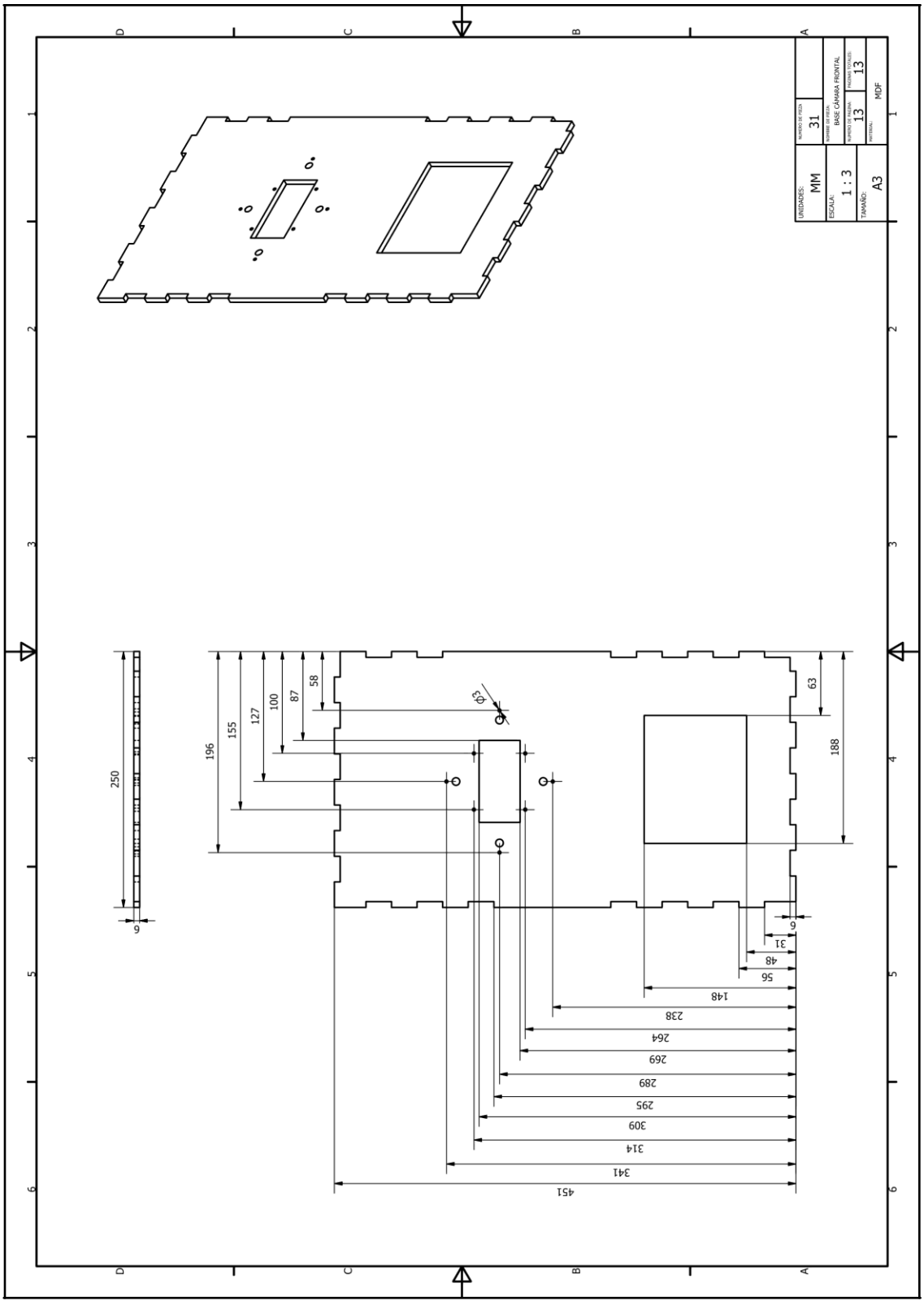

NÚMERO DE PIEZA:
18
NOMBRE DE PIEZA:
VIGA INFERIOR
NÚMERO DE PÁGINA:
8
PÁGINAS TOTALES:
13
MATERIAL:
MDF
UNIDADES:
MM
ESCALA:
1 : 2
TAMAÑO:
A4
24
45°
169
174
6
31
56
174
6


UNIDADES: MM
ESCALA: 1 : 2
TAMAÑO: A4
NUMERO DE PIEZA: 19
NOMBRE DE PIEZA: VIGA SUPERIOR
NUMERO DE PAGINA: 9
PAGINAS TOTALES: 13
MATERIAL: MDF
6
156
45°
25
50
6
31
56
156


UNIDADES:
MM
ESCALA:
1 : 1
TAMAÑO:
A4
NUMERO DE PIEZA:
21
NOMBRE DE PIEZA:
ACOPLE ARO LED
NUMERO DE PAGINA:
10
PAGINAS TOTALES:
13
MATERIAL:
ABS
104
90°
Ø4
R32
R46
R44
R34
69
R15
R5
7
3


NÚMERO DE PEÇA
23
NOME DA PEÇA
BASE CÂMARA SUPERIOR
NÚMERO DE PEÇAS
11
PEÇAS TOTAIS
13
MATERIAL
MDF
UNIDADES:
MM
ESCALA:
1 : 2
TAMANHO:
A3
250
196
167
155
127
100
87
58
25
6
9
Ø3
6
31
56
142
169
174
194
214
219
246
325


NÚMERO DE PIEZA: 28
NOMBRE DE PIEZA: ACOPLE SENSOR LÁSER
NÚMERO DE PÁGINA: 12
PÁGINAS TOTALES: 13
MATERIAL: ABS
UNIDADES: MM
ESCALA: 3 : 1
TAMAÑO: A4
31.00
27.50
6.00
18.00
3.00
14.50
11.50
9.75
3.50
9.00
ϕ3.50
5.00
10.00
13.50
17.50
24.00


NÚMERO DE PIEZA
31
NOMBRE DE PIEZA
BASE CÁMARA FRONTAL
NÚMERO DE HOJAS
13
HOJAS TOTALES
13
UNIDADES:
MM
ESCALA:
1 : 3
TAMAÑO:
A3
MATERIAL:
MDF
250
9
196
155
127
100
87
58
63
188
6
31
48
56
148
238
264
269
289
295
309
314
341
451