



**FACULTAD
DE INGENIERIA**

Universidad de Buenos Aires

MAESTRÍA EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

Lectura de fiduciales para máquina CNC

Autor:
Esp. Ing. Pablo Slavkin

Director:
MEE. Ing. Norberto M. Lerendegui (IEEE)

Jurados:
Ing. Ariel Hernandez (Seeingmachines)
Mg. Ing. Lucio Martinez (CNEA)
Dr. Daniel Minsky (CNEA/CONICET)

*Este trabajo fue realizado en la ciudad de San Carlos de Bariloche,
entre marzo de 2020 y marzo de 2021.*

Resumen

En este trabajo se describe el desarrollo de un sistema electrónico capaz de dotar de visión artificial a una máquina de control numérico de la empresa española Wolfcut y permitir el alineamiento automático de piezas mediante el reconocimiento de marcas de registro.

Se utilizó Linux en una plataforma PocketBeagle, se desarrolló un driver de *kernel* para manejar un controlador Weihong NK105, se implementó una interfaz web con HTML, JavaScript y Python, se posibilitó la transferencia remota de archivos mediante configFS, y se realizó el procesamiento de vídeo desde una cámara Wi-Fi con la biblioteca PyOpenCv.

Agradecimientos

A Juli, Valen, Maxi y León

Índice general

Resumen	I
1. Introducción general	1
1.1. Historia y principio de funcionamiento	1
1.1.1. Programa de movimientos	2
1.1.2. Controlador de movimientos	3
1.1.3. Máquina de control numérico	4
1.2. Mecanizados de piezas por control numérico	6
1.3. Software para el reconocimiento de marcas	9
1.4. Empresa interesada	10
1.5. Motivación y alcance	10
2. Introducción específica	13
2.1. Tecnologías utilizadas	13
2.2. Plataforma PocketBeagle	13
2.3. Aplicación web	14
2.4. Cámara de vídeo	16
2.5. Trigonometría de alineación	17
2.6. Detección y tipos de marcas fiduciales	20
3. Diseño e implementación	23
3.1. Diseño general del sistema	23
3.2. Bloque PocketBeagle	26
3.2.1. Teclado virtual	26
3.2.2. Pantalla virtual	28
3.2.3. Sistema virtual completo	30
3.2.4. Conexión USB como dispositivo de almacenamiento	30
3.3. Cámara de vídeo a PocketBeagle	32
3.4. Software de control	36
3.5. Software de lectura de marcas nkHACK	37
3.6. Calibracion de la camara	38
3.7. Secuencia de pasos para alinear	39
A. Ejemplos del proceso de reconocimiento de marcas	45
Bibliografía	47

Índice de figuras

1.1.	Jhon Parsons, el inventor de las máquinas de control numérico junto a una de sus máquinas.	1
1.2.	Los tres bloques básicos de una máquina CNC.	2
1.3.	Secuencia de pasos para operar una máquina NC primigenia.	3
1.4.	Secuencia de pasos para operar una máquina CNC moderna.	4
1.5.	La etapa de control se suele separar en dos: controlador lógico y driver de potencia.	4
1.6.	Esquema general y modelos de máquinas CNC dependiendo del tipo de material a procesar y la tecnología de corte.	7
1.7.	Ejemplos de piezas mecanizadas mediante máquinas CNC.	8
1.8.	Esquema de corte de una letra A en una placa de material virgen.	8
1.9.	Corte de la silueta de la letra A previamente impresa en el material.	9
1.10.	Corte de la silueta de la letra A previamente impresa en el material con lectura de marcas.	9
1.11.	Controlador NK105 de la firma Weihong elegido para extender sus funciones y dotarlo de lectura de marcas.	12
2.1.	Diagrama de bloques del sistema implementado.	13
2.2.	Conjunto de herramientas adoptadas.	14
2.3.	Capas de software relacionadas con la aplicación web utilizadas.	15
2.4.	Aplicación IP Webcam que permite utilizar la cámara del teléfono y enviar el vídeo por Wi-Fi.	17
2.5.	Corrección de desplazamiento. A representa el sistema de coordenadas de la máquina y las dimensiones extraídas del archivo de corte, B representa el objeto real desplazado, escalado y rotado respecto del primero	18
2.6.	Cálculo de rotación de la recta formada entre los puntos 1 y 2 en los planos A y B.	19
2.7.	Corrección de la rotación del plano A respecto del plano B utilizando dos puntos de referencia.	20
2.8.	Corrección de la escala entre los planos A y B utilizando como factor de escala las dimensiones relativas entre los dos planos.	20
2.9.	Ejemplo de diferentes tipos de marcas fiduciales.	21
3.1.	Comparación entre la conexión original de la máquina CNC y la modificada para el reconocimiento de marcas.	23
3.2.	Comparación entre la conexión del controlador antes y después de ser intervenido por el accesorio desarrollado.	26
3.3.	Diagrama de bloques del software de virtualización de teclado y pantalla.	27
3.4.	Diagrama de bloques implementados en el driver de <i>kernel</i> para la lectura e interpretación de las tramas de datos entre el controlador y el LCD.	29

3.5. Control de la máquina mediante la PocketBeagle. Se envían datos a una FIFO y un servicio los direcciona al controlador. Los datos del LCD son capturados por el driver y mostrados en la consola. Se puede ver la sincronía entre el mando a distancia y el virtual.	31
3.6. La PocketBeagle se comporta como un dispositivo de almacenamiento. El controlador accede al sistema de archivos en busca de trabajos a procesar.	32
3.7. Técnica de doble sistema de archivos para transferir información al controlador.	32
3.8. Soporte de celular y puntero láser solidarios al eje Z.	33
3.9. Imagen original, escala de grises y binaria obtenidas mediante un algoritmo en Python como parte del proceso de reconocimiento de marcas.	35
3.10. Proceso de filtrado y selección de perímetros dentro de la imagen. Como resultado se obtiene el centro y rotación de la marca con el área especificada.	36
3.11. Pagina de control de la maquina a traves de una pagina web.	36
3.12. Pagina de control y reconocimiento de marcas.	37
3.13. Revelacion del efecto de distorcion de las medidas en funcion del angulo de captura de la imagen, tipo de lente y distancia al objetivo.	39
3.14. Analisis de la distorcion en las medidas de imagenes tomadas en diferentes escenarios y recortadas para tomar solo una zona central de la imagen y reducir la distorcion.	39
3.15. Metodo de iteracion de movimientos para el centrado de la marca en el centro de la imagen y la mitigacion de los efectos alineales en el fotograma.	40
A.1. Secuencia de reconocimiento de marcas.	45
A.2. Secuencia de reconocimiento de marcas.	46

Índice de tablas

1.1. Modelos de controladores	5
1.2. Modelos de drivers	6
1.3. Sistemas de reconocimiento de marcas	11
2.1. Selección de la cámara	17
3.1. Características de los bloques principales	24
3.1. Características de los bloques principales. Continuación	25
3.2. Secuencia de pasos para alinear un trabajo	40
3.2. Secuencia de pasos para alinear un trabajo. Continuacion.	41
3.2. Secuencia de pasos para alinear un trabajo. Continuacion.	42
3.2. Secuencia de pasos para alinear un trabajo. Continuacion.	43
3.2. Secuencia de pasos para alinear un trabajo. Continuacion.	44

Dedicado a Fernando Sanchez

Capítulo 1

Introducción general

En el presente capítulo se explica el principio de funcionamiento de las máquinas de control numérico y las dificultades que presentan para alinearse con las piezas a mecanizar. Se exponen soluciones de alineación existentes y finalmente se comenta acerca de la motivación, el alcance y los objetivos de la propia.

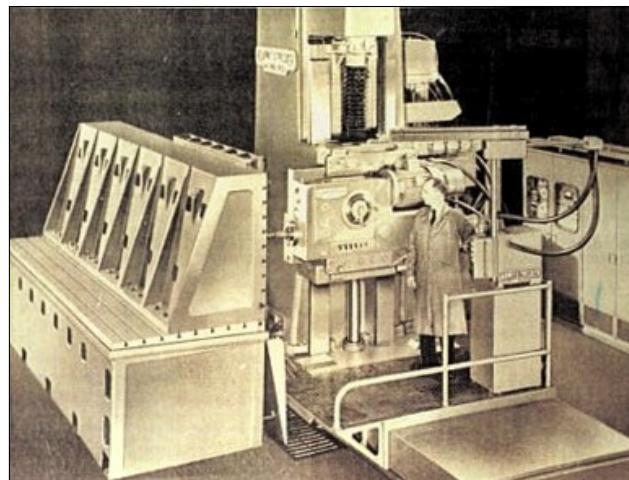
1.1. Historia y principio de funcionamiento

Hacia finales de la década del '40, el mecánico inventor Jhon Parsons¹ retratado en la figura 1.1a, logró motorizar una agujereadora de banco de precisión y la automatizó mediante el uso de una cinta perforada, figura 1.1b.

A este invento se lo considera la primera máquina de control numérico o NC por sus siglas en inglés (*numerical control*).



(A) Fotografia de Jhon T. Parsons, considerado en la industria como el inventor de las máquinas NC.



(B) Una de las primeras máquinas de la industria consideradas de control numérico NC.

FIGURA 1.1. Jhon Parsons, el inventor de las máquinas de control numérico junto a una de sus máquinas.

Luego de varias décadas, con el surgimiento de las computadoras, se reemplazaron las cintas perforadas por software, dando lugar a las máquinas de control numérico computarizado o CNC por sus siglas en inglés (*computer numerical control*).

¹https://en.wikipedia.org/wiki/John_T._Parsons

A pesar del paso del tiempo y los avances tecnológicos, los bloques constitutivos de una máquina CNC siguen siendo los mismos que se muestran en la figura 1.2.



FIGURA 1.2. Los tres bloques básicos de una máquina CNC.

1.1.1. Programa de movimientos

El programa de movimientos consiste en una serie de instrucciones necesarias para obtener una determinada pieza y se escribe en un lenguaje conocido como GCode[1].

Este lenguaje fue creado por el Instituto tecnológico de Massachusetts en la década del '50 y especificado en el documento NIST-RS274-D [2].

Originalmente los ingenieros de mecanizado lo escribían manualmente en una planilla y luego, mediante una máquina de mecanografía, se transcribía a una cinta perforada que sería interpretada por el controlador de movimientos.

Se pueden ver algunas fotos de este primigenio proceso en la figura 1.3.

En el presente se diseña la pieza en tres dimensiones con la ayuda de software de diseño asistido por computadora CAD (*computer aided design*).

Algunos software de CAD reconocidos se listan a continuación:

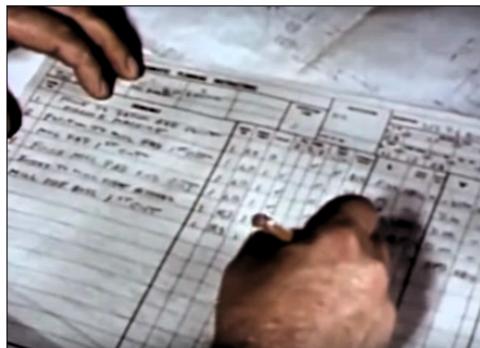
- FreeCAD
- Blender
- Rhinoceros
- AutoCAD
- SolidWorks

El archivo de salida del CAD se procesa con un software de manufactura asistida por computadora CAM (*computer aided manufacturing*) en donde el diseñador puede configurar los métodos y estrategias de mecanizado.

Algunos software de CAM reconocidos se listan a continuación:

- RhinoCAM
- Blender CAM
- FreeMILL
- Aspire
- Mastercam

El resultado final es un archivo de texto en lenguaje GCode que se almacena digitalmente y que será procesado por el controlador.



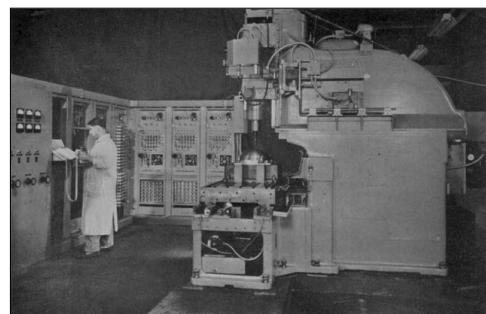
(A) Ingeniero escribiendo en papel la lista de operaciones para mecanizar una pieza en lenguaje GCode.



(B) Operadora transcribiendo la lista de operaciones del formato papel a un rollo de cinta multiperforada.



(C) Lector de cinta multiperforada que controla los movimientos de la máquina.



(D) Operador trabajando con una de las primeras fresadoras por control numérico.

FIGURA 1.3. Secuencia de pasos para operar una máquina NC pionera.

Esta secuencia es conocida como diseño CAD/CAM y se muestra en la figura 1.4

1.1.2. Controlador de movimientos

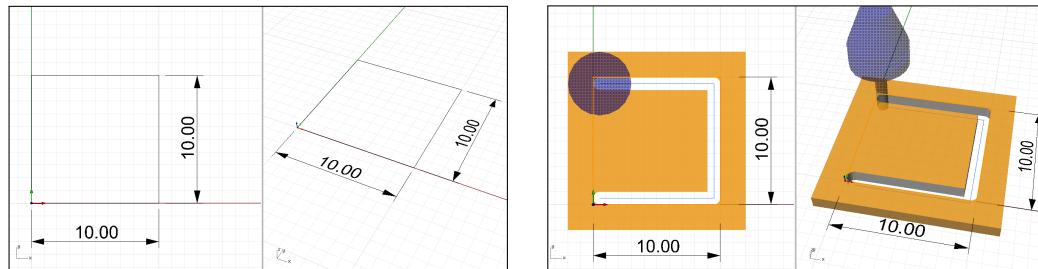
El controlador de movimientos es un equipo electrónico capaz de leer un programa en lenguaje GCode y proveer las señales eléctricas adecuadas para mover la máquina.

Es usual que a la salida del controlador se conecten amplificadores de señal (*drivers*) que proveen la potencia suficiente para mover los motores y mecanismos montados en la máquina.

De esta manera el controlador se compone de dos etapas, controlador lógico y drivers como se aprecia en la figura 1.5.

En función de la complejidad requerida para la máquina y de los requisitos de potencia para los movimientos se dimensionan el controlador y los drivers.

En las tablas 1.1 y 1.2 se listan algunos modelos de controladores y drivers comerciales detallando las características principales.



(A) Se diseña la pieza en 3D en un software CAD.

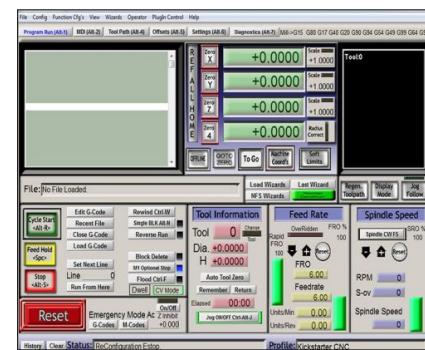
(B) Se diseña la estrategia de mecanizado y opcionalmente se simula el proceso en un software CAM.

```

1 %
2 N0001 G17 G21 G40 G90
3 N0002 T1 M06
4 N0003 S10000 M3 G4 P5 (delay 58 segs)
5 N0004 G0 X0.5000 Y0.5000 Z0.5000
6 N0005 G1 Z-1.0000 F2000.
7 N0006 X9.5000 F1500.
8 N0007 Y9.5000
9 N0008 X0.5000
10 N0009 Y0.5000
11 N0010 G0 Z0.5000
12 N0011 M30
13 %

```

(C) Se exporta desde el CAM un archivo en lenguaje GCode con las instrucciones de máquina que leerá el controlador.



(D) El software del controlador genera las señales que mueven el CNC y ejecutan el mecanizado.

FIGURA 1.4. Secuencia de pasos para operar una máquina CNC moderna.



FIGURA 1.5. La etapa de control se suele separar en dos: controlador lógico y driver de potencia.

1.1.3. Máquina de control numérico

En términos generales una máquina CNC es un conjunto de piezas electromecánicas que permiten mover el elemento de mecanizado en varias dimensiones. En algunas el elemento de mecanizado permanece fijo y lo que se mueve es la pieza a mecanizar. Suelen ser motorizadas, pero también las hay con actuadores lineales, sistemas hidráulicos o una combinación de todos estos.

Dependiendo del propósito de la máquina se definen los grados de libertad del movimiento. Es usual utilizar tres ejes perpendiculares para mesas de corte planos, seis ejes para centros de mecanizado de piezas complejas, seis para brazos roboticos y solo dos para corte y grabado de piezas planas con láser.

Para el desarrollo de este trabajo se estudian solamente máquinas de dos y tres

ejes perpendiculares, dado que la empresa interesada comercializa principalmente este tipo de estructuras.

TABLA 1.1. Modelos de controladores CNC disponibles en el mercado

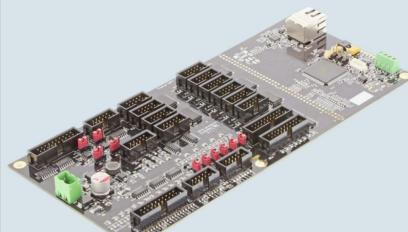
Características	Imagen
Controlador dependiente de una PC y conexión por puerto paralelo. Solución económica para máquinas de baja performance.	
Controlador integrado de media performance, ideal para máquinas profesionales pero de baja complejidad. Este es el controlador que se usará para realizar los ensayos.	
Controlador basado en PC sobre Windows de media performance. Este es el controlador que usa actualmente la empresa Wolfcut en sus máquinas.	
Controlador autónomo profesional de gran performance y opciones de operación. Este tipo de controladores se utilizan en centros de mecanizado de altísima complejidad y prestaciones sumamente exigentes. Es inusual ver este tipo de controladores en las máquinas de gama media debido a su alto costo.	

TABLA 1.2. Modelos de drivers de motores

Características	Imagen
Driver para motores paso a paso pequeños, económicos, ideales para máquinas simples, impresoras 3D, y hobby.	
Driver para motores paso a paso medianos, ideales para máquinas de media precisión y mecánica ligera.	
Driver para motores BLDC (<i>brush less direct current motors</i>), de potencia media, adecuados para máquinas de extrema precisión y escalables en potencia.	

Se esquematiza y se muestran algunos modelos de este tipo de estructuras en la figura 1.6.

1.2. Mecanizados de piezas por control numérico

En la actualidad muchos de los procesos industriales que involucran el mecanizado de piezas como las que se muestran en la figura 1.7 se realizan utilizando máquinas de control numérico computarizado o CNC [3] (*computer numerical control*).

El proceso de mecanizar piezas utilizando esta tecnología se esquematiza en la figura 1.8 y consiste en una serie de pasos como los que se enumera a continuación:

1. Posicionar la placa del material a cortar en la mesa de corte.
2. Posicionar la herramienta de corte en un punto de referencia de la placa.
3. Cargar el archivo que contiene la información de corte.
4. Cortar.

Hay casos en los cuales la placa a cortar está previamente impresa y el proceso de corte debe respetar su silueta con exactitud como se esquematiza en la figura 1.9a. Al no haberse aplicado ninguna corrección ni alineamiento entre el sistema de

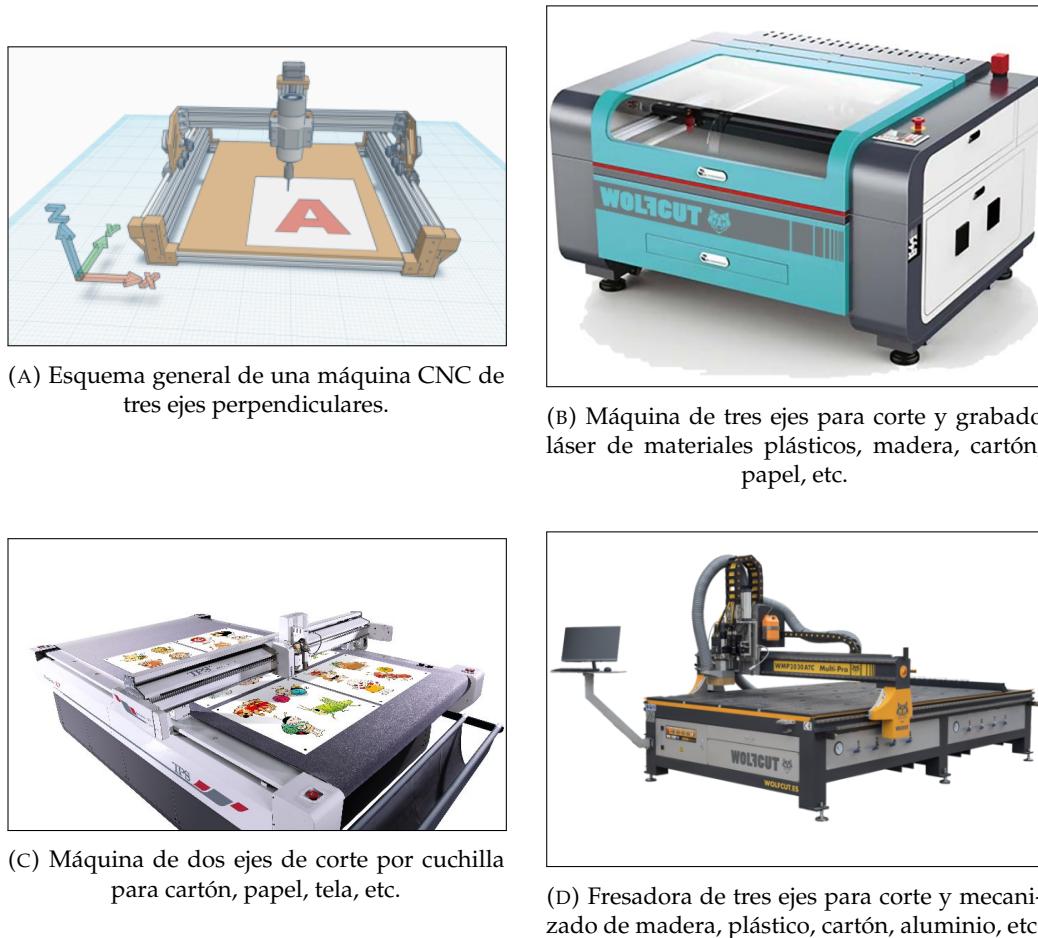


FIGURA 1.6. Esquema general y modelos de máquinas CNC dependiendo del tipo de material a procesar y la tecnología de corte.

movimientos de la máquina y la pieza, el software de corte no tiene la información de la posición, rotación y escala exacta de la pieza dispuesta en la mesa.

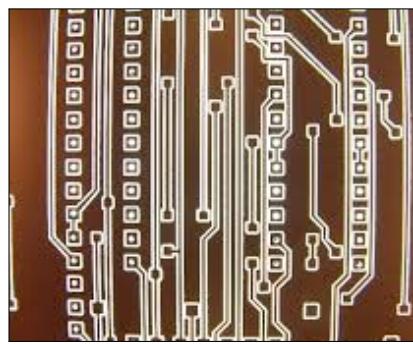
En el ejemplo mostrado en la figura 1.9b, se puede ver que la máquina no puede seguir con exactitud el contorno de la letra A.

En la industria se presenta este problema en muchos casos, algunos de los cuales se enumeran a continuación:

- Alineación de placas de circuito impreso de dos caras.
- Necesidad de volver a alinear una pieza que requiere un nuevo proceso de mecanizado.
- Necesidad de volver a alinear luego de abortar un mecanizado ante un corte de energía.
- Errores de escala y escuadra entre las diferentes máquinas involucradas en el proceso.
- Contracción y dilatación del material debido a variaciones de temperatura.
- Deformación de piezas elásticas al momento de fijarlas en la mesa de corte.



(A) Máquina CNC ejecutando el corte de piezas en madera para mobiliarios.



(B) Placa de circuito impreso realizado mediante el fresado del contorno de sus pistas.

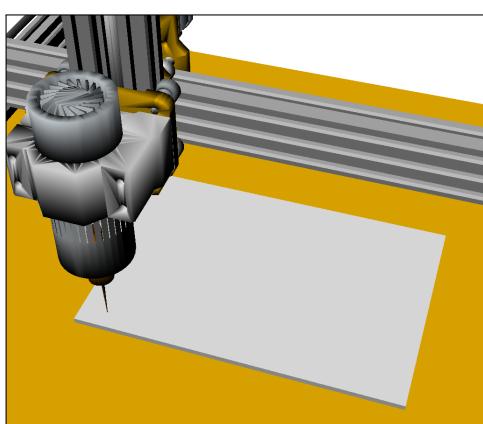


(C) Corte y fresado en placa de aluminio para obtener un repuesto de una máquina herramienta.

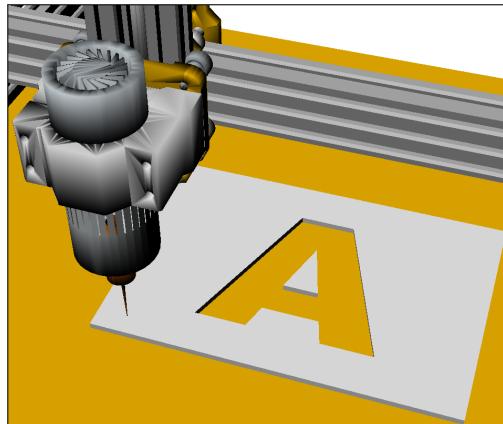


(D) Corte de logos y letras en madera.

FIGURA 1.7. Ejemplos de piezas mecanizadas mediante máquinas CNC.



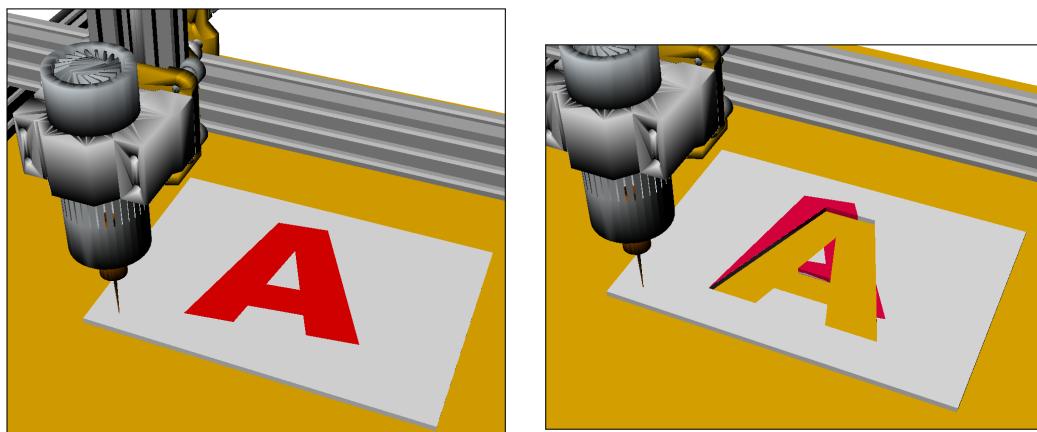
(A) Placa a cortar fijada a la mesa de corte y fresa de corte en posición.



(B) Pieza cortada.

FIGURA 1.8. Esquema de corte de una letra A en una placa de material virgen.

En el presente trabajo se aplican técnicas de visión artificial para reconocer los puntos de referencia que permiten corregir esta desalineación. Estos puntos se incluyen en el proceso de diseño y se imprimen junto con el trabajo a mecanizar.

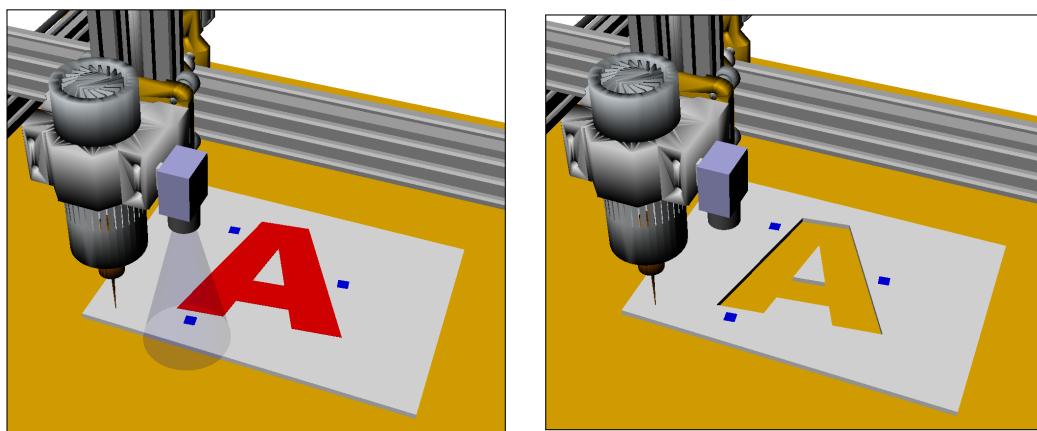


(A) Placa a cortar fijada a la mesa de corte y fresa de corte en posición.

(B) Pieza cortada con una notable desalineación entre la silueta previamente impresa y el corte.

FIGURA 1.9. Corte de la silueta de la letra A previamente impresa en el material.

Mediante el uso de una cámara de vídeo montada en el CNC se puede corregir el desplazamiento, el ángulo y la escala del objeto impreso en relación al sistema de coordenadas de la máquina. El resultado esperado se muestra en la figura 1.10.



(A) Placa a cortar impresa con la letra A y tres marcas. Se encuentra fijada a la mesa con la fresa de corte en posición.

(B) Alineación del trabajo de corte según las marcas azules de referencia. La pieza se muestra mecanizada siguiendo el contorno sin errores.

FIGURA 1.10. Corte de la silueta de la letra A previamente impresa en el material con lectura de marcas.

1.3. Software para el reconocimiento de marcas

En la tabla 1.3 se destacan algunos desarrollos de software que permiten extenderse o adaptarse para soportar el reconocimiento de marcas. Se puede ver que la mayoría de las soluciones del mercado están basadas en PC y eso aumenta el costo general del sistema, disminuye la fiabilidad y limita el acceso desde múltiples plataformas.

Se ha podido constatar que además del costo de hardware, licencias o extensiones de software, el costo de las cámaras utilizadas son privativos para el segmento de máquinas de gama baja o media que es el segmento principal de este trabajo. La empresa interesada, Wolfcut², utiliza actualmente uno de estos sistemas con algunos resultados adversos.

No se ha encontrado ningún sistema que utilice una cámara con conexión inalámbrica, y tampoco aprovechando el uso de la cámara de un teléfono celular.

Por otra parte tampoco se ha encontrado alguna solución completa de código abierto y colaborativo que facilite el crecimiento del proyecto con la ayuda de la extensa comunidad de usuarios, desarrolladores y entusiastas.

1.4. Empresa interesada

Este trabajo se realiza en el marco de una colaboración con la empresa española Wolfcut, la cual fabrica y comercializa máquinas de control numérico en toda Europa.

Se pueden ver algunos de sus productos en su página web <https://wolfcut.es/>.

Cuenta con más de 20 años de experiencia en el rubro y algunas innovaciones en el área de reconocimiento de marcas.

Sin embargo las soluciones que ha utilizado hasta el momento requieren el uso de una PC con Windows para su funcionamiento y la empresa considera que estas tecnologías no son adecuadas para el ámbito industrial de sus clientes.

Es por ello que se trabajó en conjunto para lograr una solución embebida.

1.5. Motivación y alcance

La principal motivación de este trabajo es lograr extender las capacidades de un controlador embebido de uso profesional y dotarlo de visión artificial para el reconocimiento de marcas fiduciales.

Con los argumentos y la experiencia de la empresa Wolfcut, se determinó que uno de los controladores de uso profesional más popular del mercado es el NK105 de la firma Weihong [11] que se muestra en la figura 1.11.

Este controlador solo cuenta con un comando remoto para todas las operaciones de manejo y configuración.

No provee una API definida por el fabricante ni una interfase física para poder conectarse y extender sus funciones.

Por otro lado, al estar basado internamente en una FPGA y no depender de una PC para funcionar, es reconocido por sus excelentes resultados de corte, su estabilidad en trabajos extensos, una excelente fiabilidad, y un costo muy accesible.

Las capacidades de operación del NK105 son relativamente simples, pero de nivel profesional con un mercado ya consolidado y muy extenso en todo el mundo.

²<https://wolfcut.es/>

TABLA 1.3. Se destacan algunos modelos y marcas de sistemas de reconocimiento de marcas y sus características principales.

Características	Imagen
EddingCNC [4]: Software basado en PC sobre Windows al cual varios fabricantes como GES [5] y Wolfcut [6] lo han extendido para soportar reconocimiento de marcas.	
myCNC [7]: Esta aplicación de la empresa pv-automation [8] ofrece un sistema de visión artificial y reconocimiento de marcas basado en una PC industrial y cámaras USB.	
Machinekit [9]: Es un software de control que opera sobre Linux al cual se le han hecho algunas intervenciones para el reconocimiento de marcas.	
Summa [10]: Es una línea de máquinas de corte de contornos, principalmente en papel, que cuenta con lectura de marcas integrado en sus sistemas embebidos.	

Además este modelo pertenece a una familia de soluciones de complejidad creciente, pero que comparten el controlador. Las soluciones que se le apliquen se pueden extender a toda la familia.

El alcance de este trabajo se limita a intervenir y dotar de lectura de marcas al controlador NK105 y obtener resultados comparables con otras soluciones de mercado.



FIGURA 1.11. Controlador NK105 de la firma Weihong elegido para extender sus funciones y dotarlo de lectura de marcas.

Capítulo 2

Introducción específica

En el presente capítulo se introducen las tecnologías más relevantes involucradas, se explica el álgebra y la geometría asociada a la alineación de dos planos y finalmente los criterios y selección de marcas de registro.

2.1. Tecnologías utilizadas

En el diagrama de bloques de la figura 2.1 se muestran las partes que componen el sistema implementado. Para explicar como se relacionan e interactúan entre sí estos bloques, en las próximas secciones de esta memoria se describen las cualidades y funciones más destacadas de cada uno.

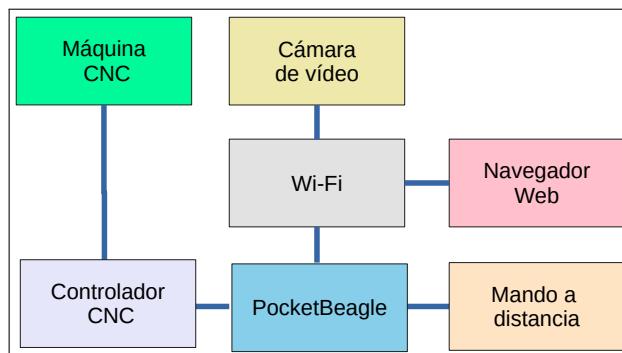


FIGURA 2.1. Diagrama de bloques del sistema implementado.

2.2. Plataforma PocketBeagle

PocketBeagle es un miembro del ecosistema de plataformas de desarrollo BeagleBoard. [12]. Las características de esta plataforma, que se muestra en la figura 2.2.A, y que son relevantes para este trabajo son las siguientes:

- Controlador integrado SiP (system-in-package) Octavo Systems OSD3358-SM.
- Memoria de 512MB DDR3.
- Unidad de procesamiento de 32b Cortex-A8 @1-GHz.
- 72 pines de expansión, UART, SPI, I2C, entre otras.
- USB de alta velocidad.

Durante la carrera de maestría se obtuvo experiencia en el uso de otra plataforma de la misma familia, la BeagleBoneBlack¹, sobre la cual se corrió un sistema operativo Linux y se desarrollaron drivers para manejar interfaces de comunicación. Dicha experiencia permitió argumentar que la PocketBeagle cuenta con las interfaces de comunicación necesarias y es capaz de correr el software requerido para este trabajo a una fracción del costo y tamaño.

La única falencia es que no cuenta con una interfaz Wi-Fi ni Ethernet pero se resolvió utilizando un adaptador USB como se destaca en la figura 2.2b.

Se está utilizando una distribución oficial del sistema operativo Debian compilada para esta plataforma y que se puede descargar desde el link oficial².

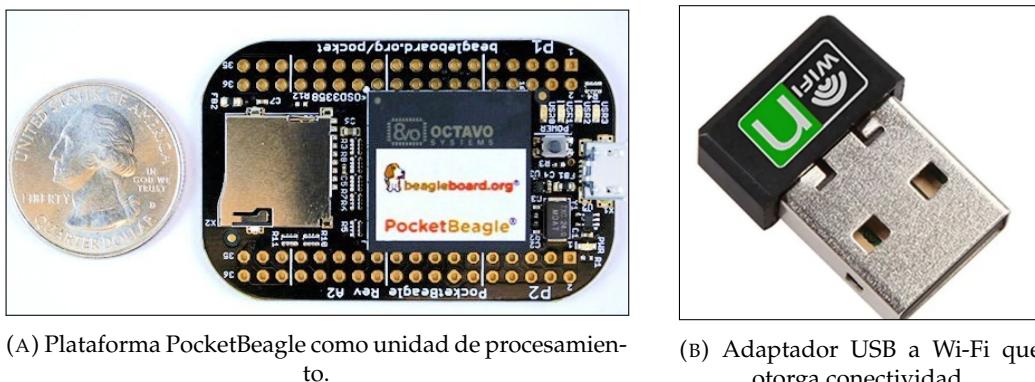


FIGURA 2.2. Conjunto de herramientas adoptadas.

Se evaluaron plataformas más potentes como la PYNQ-Z1 [13]. Si bien las prestaciones son mayores a la PocketBeagle, también lo es el costo, y dado que este trabajo es solo un accesorio para un controlador de máquinas de segmento medio y bajo, se intentó mantener los costos y la complejidad acotada.

2.3. Aplicación web

La interfaz de usuario se desarrolló utilizando tecnologías web para permitir acceder desde cualquier dispositivo con un navegador web.

Los argumentos a favor de esta tecnología están basados en la falta de aplicaciones para el manejo de máquinas CNC que sean independientes del sistema operativo del ordenador del cliente.

Muchos usuarios utilizan herramientas de diseño sobre el sistema operativo macOS³, y deben contar con un segundo ordenador para poder interactuar con el CNC.

Utilizando la tecnología web, solo basta con abrir un navegador desde el mismo entorno de trabajo para operar el CNC.

Para cumplir con los requisitos planteados se utilizó un arreglo de tecnologías web muy variadas, pero muy ligadas entre sí, que se muestran en la figura 2.3.

¹<https://beagleboard.org/black>

²<https://beagleboard.org/latest-images>

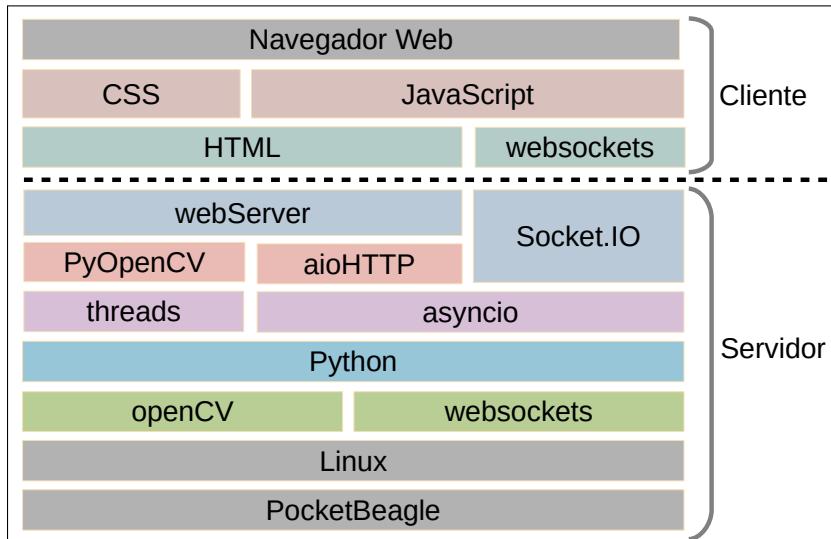


FIGURA 2.3. Capas de software relacionadas con la aplicación web utilizadas.

Para entender la función principal de cada capa se describen algunos detalles en la siguiente lista:

- Python [14]: Es un poderoso y popular lenguaje de programación en el cual se corre principalmente el servidor web, y las funciones de procesamiento de imágenes. La imagen del sistema operativo de la PocketBeagle ya cuenta con Python versión 3 instalado por defecto, lo que asegura compatibilidad.
- asyncio [15]: Es una biblioteca para Python que permite correr una única tarea que administra muchas de manera concurrente y cooperativa. Esto permite que, por ejemplo, una función de Python esté esperando datos de un archivo mientras otra procesa una imagen sin bloquear las funciones del servidor.
- aiohttp [16]: Es una biblioteca de Python que permite correr un servidor web utilizando la infraestructura de asyncio para realizar tareas de manera cooperativa y concurrente. Es el motor del servidor web.
- HTML5.0 [17]: Es el lenguaje de marcas utilizado para visualizar contenidos en la web. Se utiliza para mostrar contenido estático y también para aprovechar un mecanismo nativo de la versión 5.0 que permite la reproducción de vídeo en la página web sin necesidad de otras tecnologías.
- CSS [18]: Es un lenguaje que permite definir estilos, colores, formato y modo de presentación en pantalla de una página escrita en HTML. Es indispensable para crear aplicaciones web atractivas y apropiadas para cada uso.
- JavaScript [19]: Es un lenguaje de programación intrínsecamente relacionado con HTML que permite la creación de páginas web dinámicas. La mayoría de los navegadores modernos soportan este lenguaje y esto permite que la aplicación que se desarrolla pueda correr en cualquier plataforma que cuente con un navegador web. Más del 90 % de la aplicación web desarrollada está codificada en lenguaje JavaScript: los cálculos de alineación,

la interacción con el usuario, etc. También se utilizan bibliotecas de terceros para diferentes usos escritas en este lenguaje, lo que permite reutilizar código.

- WebGL [20]: Es una biblioteca gráfica (*Web graphics library*) escrita en JavaScript, que permite definir y visualizar objetos en tres dimensiones en una página web. Esta íntimamente ligada con el desarrollo web y es por ello que puede aprovechar las tarjetas gráficas del ordenador del cliente para acelerar las tareas de visualización y procesamiento. De esta manera logra eficiencias similares a las aplicaciones nativas del sistema operativo.
- Three.js [21]: Es una biblioteca escrita en JavaScript, que utiliza la tecnología WebGL y facilita la creación de objetos, cuenta con muchos ejemplos y casos de uso, abstrae al programador de los detalles de implementación y mejora el mantenimiento del código. Se utiliza en la aplicación para visualizar el movimiento de la máquina en 3D, los trazos de corte, las correcciones de rotación, etc.
- Websockets [22]: Es un protocolo de comunicaciones que opera sobre el protocolo TCP/IP, similar a HTTP, pero diseñado con la premisa de lograr una comunicación bidireccional de baja latencia. Es de gran importancia en la aplicación para lograr una rápida respuesta de operación.
- socket.IO [23]: Es una biblioteca de JavaScript que utiliza Websockets para permitir la comunicación bidireccional entre el servidor web y el o los clientes. Toda la comunicación entre los scripts de JavaScript que corren en el cliente y el servidor en Python que corre en la PocketBeagle se comunican utilizando esta biblioteca.
- OpenCV [24]: Es una biblioteca muy popular escrita en C++ para procesamiento de imágenes asistido por computadora. Además de contar con potentes algoritmos de procesamiento muy útiles para este trabajo, es soportada por muchas plataformas asegurando la compatibilidad entre dispositivos. En particular la PocketBeagle utiliza la biblioteca libopencv-dev extraída de los repositorios oficiales de Debian.
- PyOpenCV [25]: Es una biblioteca de Python que permite utilizar las funciones de OpenCV. Dado que este trabajo está escrito en Python, se utiliza esta biblioteca para el procesamiento de marcas, la cual a su vez utiliza libopencv-dev del sistema operativo.

2.4. Cámara de vídeo

Los criterios para la selección de la cámara de vídeo se basaron principalmente en la interfaz de comunicación, los costos, la calidad de imagen y la facilidad de adquisición en mercado local. Con dichos criterios se confeccionó la tabla 2.1 con las opciones más destacadas calificando de 0 a 5.

Según esta tabla, la mejor opción es el microscopio USB, son económicos y fáciles de conseguir en el mercado local. Permiten ajustar el área de visualización al tamaño de las marcas.

Sin embargo la distancia entre el controlador de la máquina y la cámara es mayor a la que soporta el canal USB. Para poder utilizarlo de manera confiable se

TABLA 2.1. Tabla comparativa entre diferentes cámaras calificadas de 0 a 5.

Tipo	Interfaz	Calidad de imagen [0-5]	Disponibilidad [0-5]	Costo [0-5]
Celular	Wi-Fi	4	5	2
Microscopio	USB	5	5	5
web-cam	USB	3	5	3
Industrial	Ethernet	5	1	1

requiere de un amplificador que permita extender su alcance. Esto aumenta el costo total de la solución y agrega elementos susceptibles de fallas.

Por lo tanto para el alcance de este trabajo se optó por usar la cámara de un teléfono celular en conjunto con la aplicación IP Webcam [26] para la transmisión de video como se muestra en la figura 2.4.

Esta opción resuelve el problema de la longitud del cable dado que transmite por Wi-Fi, pero también permite utilizar varios modelos de teléfonos simultáneamente sin cambiar el software y poder tomar imágenes desde diferentes ángulos.

Esta aplicación cuenta además con una interfaz web desde la cual se pueden ajustar los parámetros de la cámara más importantes: zoom, brillo, desplazamiento, resolución y calidad de imagen.

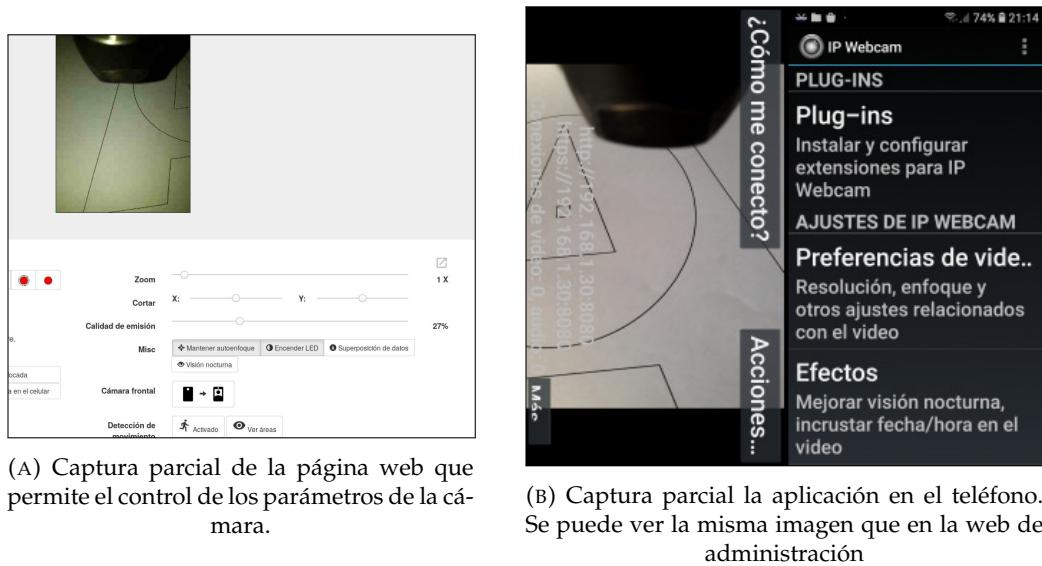


FIGURA 2.4. Aplicación IP Webcam que permite utilizar la cámara del teléfono y enviar el vídeo por Wi-Fi.

2.5. Trigonometría de alineación

El objetivo del método es poder conocer la dimensión, la posición y la rotación de un objeto relativo a la máquina.

Como la pieza que se desea mecanizar y también la propia mecánica de la máquina podrían estar distorsionadas, lo que importa es solo su relación.

Como se trata de una alineación en dos dimensiones, en geometría implica posicionar, escalar y rotar un plano respecto de otro.

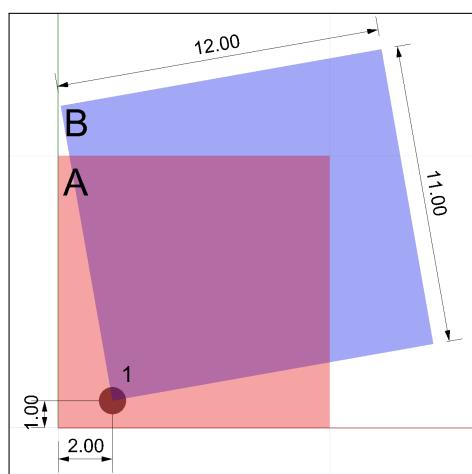
Dado dos planos A y B superpuestos como se muestra en la figura 2.5a para el siguiente análisis se define al plano A, en rojo, como el sistema de coordenadas de la máquina y sus dimensiones las establecidas en el archivo de corte. Mientras que B, en azul, como el objeto real a mecanizar que se encuentra desplazado, rotado y escalado respecto al primero.

Conociendo las coordenadas de un solo punto en los dos sistemas de referencia, se puede establecer el desplazamiento y corregirlo como se realiza en la figura 2.5b.

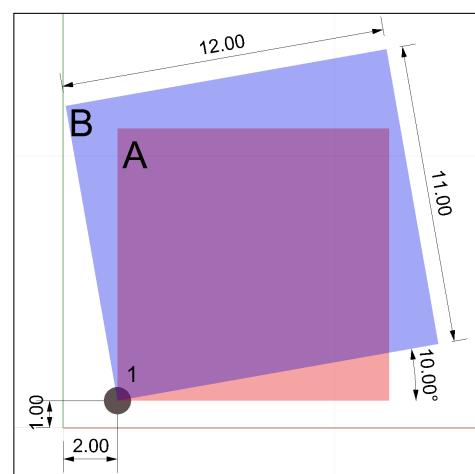
El punto 1 en el sistema A es el (2, 1) mientras que en el sistema B es el (0, 0).

La ecuación que corrige la posición del plano A es la 2.1

$$\begin{aligned} A_x(x) &= x + x_{1B} \\ A_y(y) &= y + y_{1B} \end{aligned} \quad (2.1)$$



(A) El plano B se encuentra desplazado 2 unidades en el eje x y 1 unidad en el eje Y.



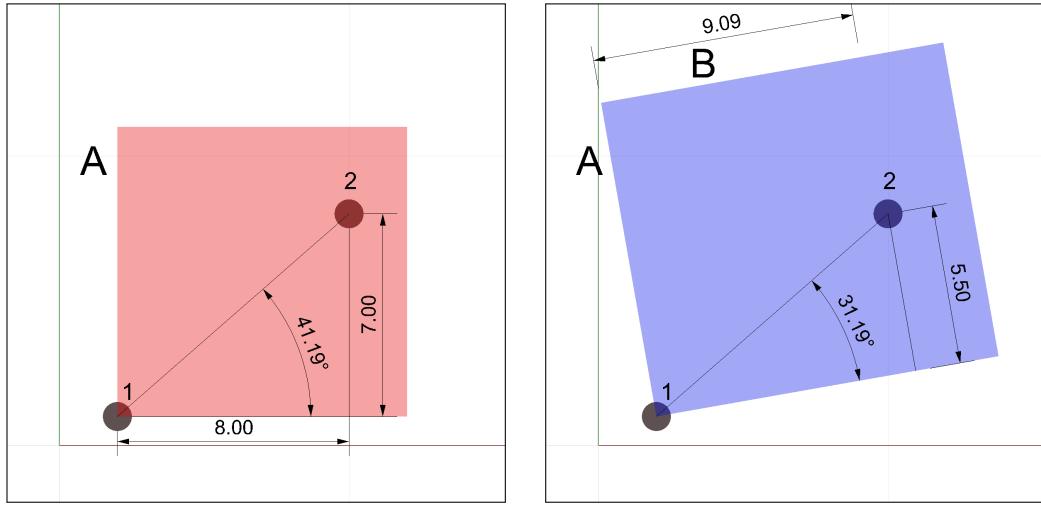
(B) El plano A se desplaza y corrige su posición.

FIGURA 2.5. Corrección de desplazamiento. A representa el sistema de coordenadas de la máquina y las dimensiones extraídas del archivo de corte, B representa el objeto real desplazado, escalado y rotado respecto del primero

Ahora, si se considera el punto 2, como se muestra en la figura 2.6, se puede calcular la rotación relativa entre B y A.

Como primer paso, aplicando trigonometría se calcula el ángulo que forma el punto 2 con el punto 1 en el plano A, luego el ángulo del punto 2 con el punto 1 pero en coordenadas del plano B. Su diferencia es la rotación del plano A respecto al plano B.

Se puede ver gráficamente en las figuras 2.6a y 2.6b y se expresa en las ecuaciones 2.2.



(A) Se calcula el ángulo de la recta formada por los puntos 1 y 2 en el plano A.
 (B) Se calcula el ángulo de la recta formada por los puntos 1 y 2 en el plano B.

FIGURA 2.6. Cálculo de rotación de la recta formada entre los puntos 1 y 2 en los planos A y B.

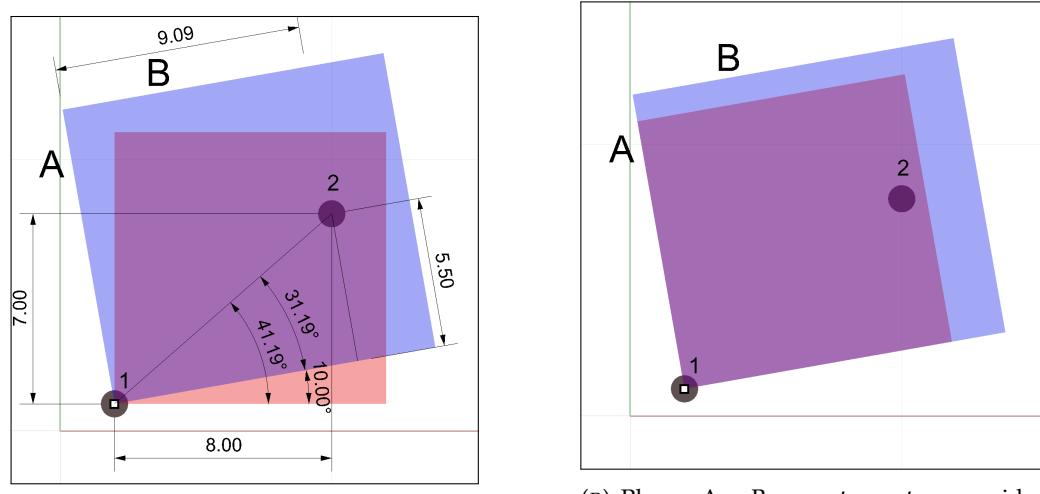
$$\begin{aligned}
 R_A &= \arctan\left(\frac{x_{2A}}{y_{2A}}\right) \\
 &= \arctan\left(\frac{7}{8}\right) \\
 &= 41.18^\circ \\
 R_B &= \arctan\left(\frac{x_{2B}}{y_{2B}}\right) \\
 &= \arctan\left(\frac{5,5}{9,09}\right) \\
 &= 31.18^\circ \\
 R_{AB} &= R_A - R_B \\
 &= 10^\circ
 \end{aligned} \tag{2.2}$$

Una vez obtenida la diferencia de ángulos se corrige rotando el plano A respecto del B como se muestra en la figura 2.7.

Para completar el proceso y conseguir la alineación final resta escalar el plano A relacionando las dimensiones con el plano B.

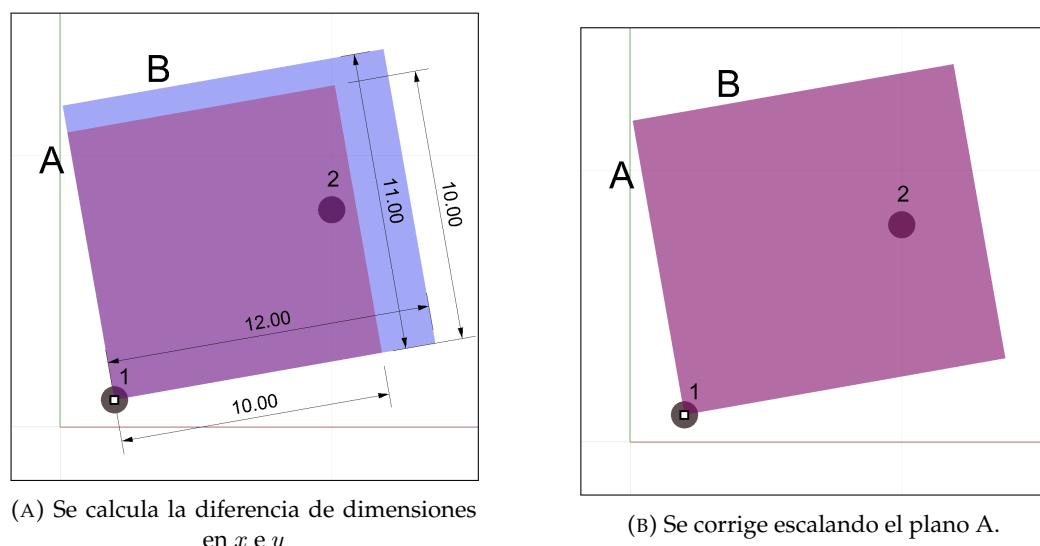
Se muestra gráficamente esta corrección en la figura 2.8 y se expresa en las ecuaciones 2.3.

$$\begin{aligned}
 S_{Ax} &= \frac{x_B}{x_A} \\
 &= \frac{12}{10} \\
 &= 1,2 \\
 S_{Ay} &= \frac{y_B}{y_A} = \frac{11}{10} \\
 &= 1,1
 \end{aligned} \tag{2.3}$$



(A) Superposición de los ángulos de la recta formada por los puntos 1 y 2.

(B) Planos A y B correctamente corregidos según la diferencia de los ángulos calculada



(A) Se calcula la diferencia de dimensiones en x e y

(B) Se corrige escalando el plano A.

FIGURA 2.8. Corrección de la escala entre los planos A y B utilizando como factor de escala las dimensiones relativas entre los dos planos.

2.6. Detección y tipos de marcas fiduciales

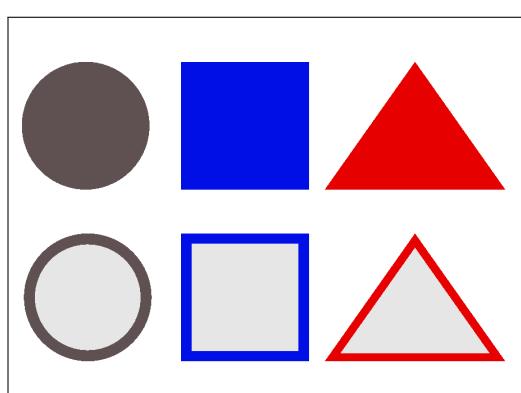
Es posible utilizar diversas figuras geométricas e incluso colores para identificar la posición de una marca, sin embargo en el mercado gráfico y de mecanizado es usual encontrar que las marcas son círculos o cuadrados de entre 1mm y 10mm de diámetro o de lado.

Para un efectivo reconocimiento de una marca utilizando una cámara de vídeo algunas consideraciones deben tenerse en cuenta como ser:

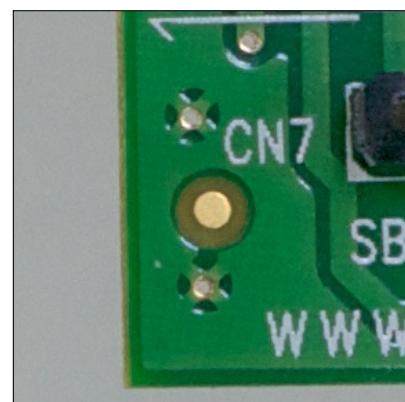
- Maximizar el contraste entre el fondo del objeto y la marca.
- Evitar irregularidades en el trazado del contorno.

- De ser posible que la marca esté pintada internamente y que no sea solo un contorno.
- Que esté alejada de bordes y otras figuras de la pieza.
- En el caso de marcas cuadradas es posible calcular el centro y también una aproximación del ángulo.
- En el caso de marcas circulares se logra mejor precisión en la detección del centro pero dado su simetría radial no se cuenta con la información de rotación.

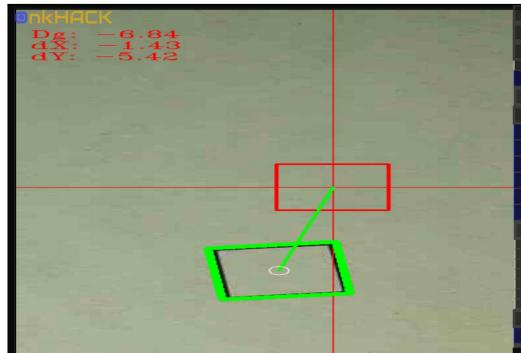
Algunos ejemplos de marcas fiduciales se muestran en la figura 2.9.



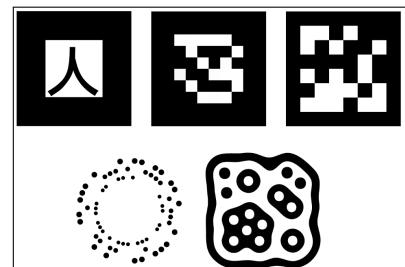
(A) Figuras geométricas llenas o contorno



(B) Ejemplo de marcas fiduciales en una placa de circuito impreso PCB.



(C) Reconocimiento de un contorno cuadrado utilizando el software desarrollado.



(D) Marcas fiduciales codificadas. Se pueden utilizar para reconocer la posición y un código para diversos usos.

FIGURA 2.9. Ejemplo de diferentes tipos de marcas fiduciales.

Para el alcance de este trabajo solo se utilizarán marcas geométricas llenas o sus contornos.

Capítulo 3

Diseño e implementación

En este capítulo se desglosan los bloques principales del sistema, se explica su interconexión, se exponen los criterios de diseño y se destacan los aspectos más relevantes de cada implementación.

3.1. Diseño general del sistema

En los diagramas de bloques de la figura 3.1 se realiza la comparación entre la topología original, que se muestra en la figura 3.1a y el modificado para el reconocimiento de marcas, representado en la figura 3.1b.

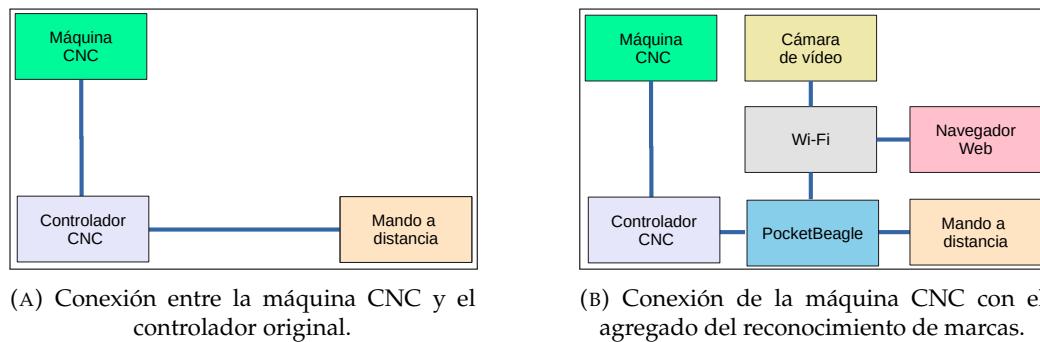


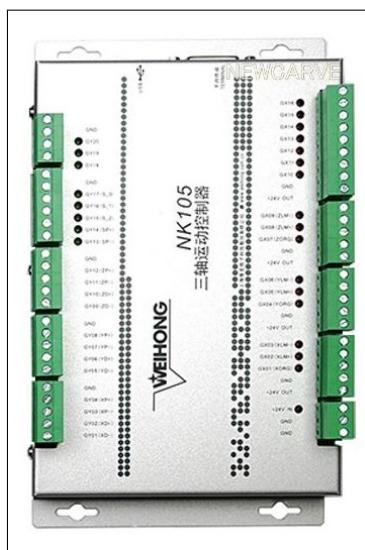
FIGURA 3.1. Comparación entre la conexión original de la máquina CNC y la modificada para el reconocimiento de marcas.

En la tabla 3.2 se detallan las principales características de cada bloque junto a una imagen representativa:

TABLA 3.1. Descripción e imagen representativa de los bloques principales del sistema.

Características del bloque	Imagen
<p>Bloque Wi-Fi: representa una red inalámbrica compartida entre los bloques “PocketBeagle”, “cámara de vídeo” y “navegador web”. Si además contara con acceso a internet, permitiría acceder remotamente.</p>	
<p>Bloque navegador web: representa un dispositivo que puede correr un navegador web: ordenador personal, teléfono celular, tableta, etc. Sin embargo para el alcance de este desarrollo se sugieren pantallas mayores a 10”.</p>	
<p>Bloque PocketBeagle: representa la plataforma de desarrollo elegida con el agregado de una ciruitería que permite intercalarse entre los bloques “controlador CNC” y “mando a distancia”.</p>	
<p>Bloque máquina CNC: representa el conjunto de piezas electromecánicas motorizadas para realizar el mecanizado de piezas.</p>	

TABLA 3.1. Descripción e imagen representativa de los bloques principales del sistema. Continuación.

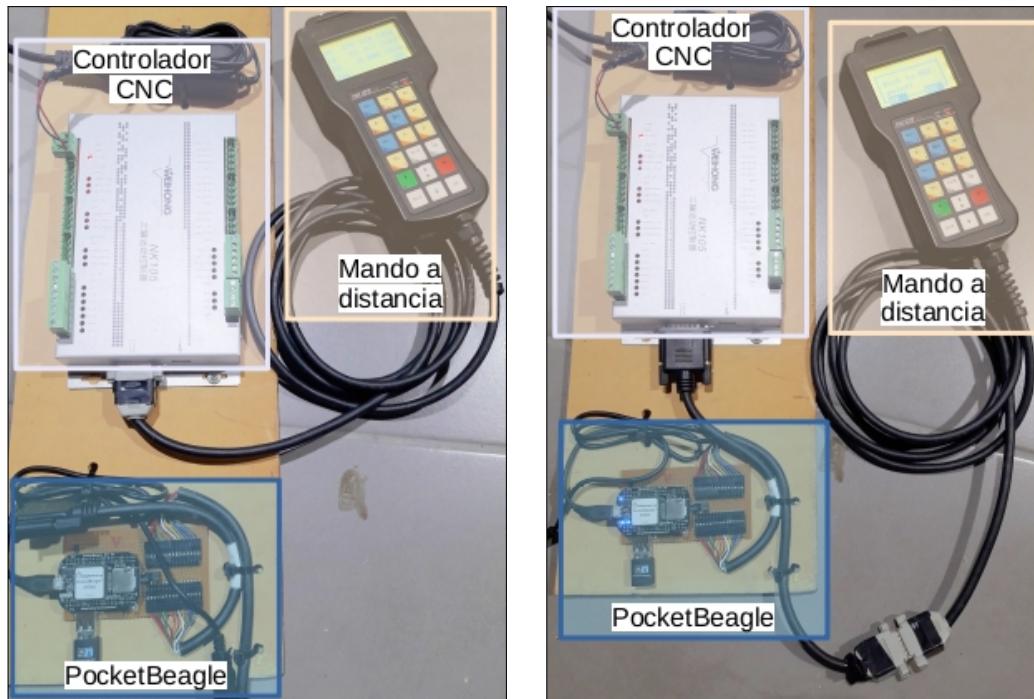
Características del bloque	Imagen
<p>Bloque controlador CNC: representa el dispositivo electrónico encargado de comandar los motores de la máquina para generar los movimientos de corte.</p>	
<p>Bloque mando a distancia: representa el dispositivo electrónico que se conecta al controlador. Cuenta con una pantalla para visualización y un teclado para el ingreso de datos.</p>	
<p>Bloque cámara de vídeo: representa el dispositivo que se monta en la máquina CNC y permite capturar las imágenes que se utilizarán para el reconocimiento de marcas. En esta implementación se utiliza la cámara de un teléfono celular.</p>	

3.2. Bloque PocketBeagle

Una de las funciones de este bloque es interceptar las señales entre el controlador y el mando para poder intervenirlas. Se estudió en detalle el cable que los interconecta y se determinó que conduce dos canales de comunicación que se detallan en la siguiente lista:

- **UART:** *universal asynchronous receiver-transmitter* es una comunicación serial que comunica periódicamente el estado del teclado al controlador.
- **SPI:** *synchronous peripheral interface* es una interfaz sobre la cual se envían los datos desde el controlador a la pantalla de cristal líquido o LCD *liquid crystal display*.

Para poder interactuar con estas interfaces se intercaló una tarjeta electrónica que las conecta a la plataforma PocketBeagle. En la figura 3.2 se muestra esta conexión en comparación con la original.



(A) Controlador CNC conectado con el mando.

(B) Controlador CNC conectado con el mando pero intervenido por el bloque “PocketBeagle”.

FIGURA 3.2. Comparación entre la conexión del controlador antes y después de ser intervenido por el accesoario desarrollado.

Como la conexión entre el mando a distancia y el controlador se realiza con un conector, el proceso de intercalar el bloque “PocketBeagle” es un proceso reversible, no modifica la funcionalidad original del sistema y permite instalarse sin dificultades.

3.2.1. Teclado virtual

Para lograr enviar datos desde la PocketBeagle al controlador se desarrolló una aplicación en lenguaje C que corre como servicio. En la figura 3.3 se muestra el diagrama de bloques del software.

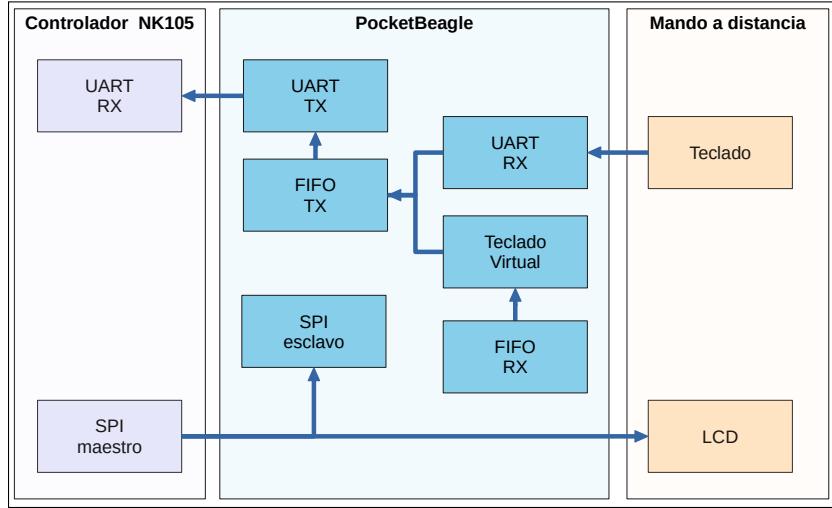


FIGURA 3.3. Diagrama de bloques del software de virtualización de teclado y pantalla.

En los fragmentos de código 3.1, 3.2 y 3.3 se pueden ver tres tareas o *threads* que implementan: la lectura del teclado físico, la lectura de una cola virtual y el envío de los datos al controlador respectivamente.

```

1 void* rcvFunc(void* niyto)
2 {
3     char frame[FRAME_SIZE];
4     while(true) {
5         while ( Get_Byt(PORT_NUMBER, frame )<1 || frame[0] != FRAME_HEADER)
6             ;
7         PollComport(PORT_NUMBER, frame+1, FRAME_SIZE-1);
8         if(memcmp(frame, FRAME_DEFAULT, FRAME_SIZE))
9             mq_send(msgQueue, frame, FRAME_SIZE, 1);
10    }
11 }
12 }
```

CÓDIGO 3.1. Tarea encargada de procesar los datos del teclado físico y reenviarlos a la cola de multiplexado.

Se aprovecharon los mecanismos de colas o FIFO's *first in first out* que ofrece Linux para multiplexar los datos del "teclado" con un nuevo bloque: "teclado virtual".

Con esta técnica, si se envían datos al bloque "FIFO Rx" los datos se reenvían al controlador emulando el presionado de un botón y al mismo tiempo se atiende de la comunicación del teclado original. De esta manera el controlador se puede manejar virtual o físicamente.

Dado que la recepción es a través de una FIFO, es posible instanciar más de un teclado virtual con accesos concurrentes [27].

```

1 void* recvVirtual(void* niyto)
2 {
3     char buf [ MAX_VIRTUAL_CMD_LENGTH ];
4     char frame[ FRAME_SIZE ];
5     FILE* pipeVi;
6     while(pipeVi=fopen(PIPE_VI,"r")) {
7         while(fgets(buf,MAX_VIRTUAL_CMD_LENGTH,pipeVi)>0) {
8             memcpy(frame,FRAME_DEFAULT,FRAME_SIZE);
9             mapBtn2Bits(atoi(buf),frame);
10            crc(frame);
11            if(memcmp(frame,FRAME_DEFAULT,FRAME_SIZE)) {
12                mq_send(msgQueue,frame,FRAME_SIZE,1);
13            }
14        }
15        fclose(pipeVi);
16    }
17 }
18

```

CÓDIGO 3.2. Tarea que recibe datos del teclado virtual y los reenvía a la cola de multiplexado.

```

1 void* sendFunc(void* niyto)
2 {
3     struct timespec tm;
4     char frame[ FRAME_SIZE ];
5     while ( true ) {
6         clock_gettime(CLOCK_REALTIME, &tm);
7         tm=timespec_add (tm,(struct timespec){0, QUEUE_SND_TOUT});
8         mq_timedreceive( msgQueue,frame,FRAME_SIZE,NULL,&tm);
9         sendBuf      ( PORT_NUMBER ,ans?frame:FRAME_DEFAULT ,
10           FRAME_SIZE );
11     }
12 }
13

```

CÓDIGO 3.3. Tarea de multiplexado de los datos del teclado físico y virtual.

3.2.2. Pantalla virtual

Para poder conocer el estado del controlador es necesario contar con la información que se muestra en la pantalla LCD.

Se estudió en detalle las especificaciones del controlador de esta pantalla con el objetivo de emular una pantalla virtual.

Debido a la alta velocidad de los datos y que las tramas no tienen un largo definido, no fue posible utilizar los drivers de SPI del *kernel* y realizar esta emulación en espacio de usuario.

Para poder capturar correctamente estas tramas se implementó un driver SPI en modo esclavo *SPI slave* como un nuevo módulo del sistema operativo [28]. Este driver, a diferencia del original, permite recibir cualquier longitud de trama, en cualquier momento y almacenarla en un espacio de memoria contigua. Esto se logró utilizando las siguientes técnicas:

- Interrupciones: se utilizó una interrupción conectada a la señal de selección de chip *CS chip select*. Esto permite reaccionar rápidamente cuando se inicia una transacción.
- Acceso directo a memoria: se utilizó el subsistema DMA *direct access memory* para que las operaciones de copia de los datos que se reciben por SPI a memoria se realicen sin la intervención del procesador. Esto permite conservar los recursos del procesador para otras tareas.
- Comunicación interprocesos: se utilizaron diversos métodos de IPC's *inter-process communications* que ofrece Linux para lograr un sistema reactivo con mínimo retardo [28].
- Operaciones de archivo: para transferir las tramas decodificadas del espacio de *kernel* al de usuario, se implementó un acceso al módulo como un archivo virtual. De esta manera se aprovechan las herramientas de lectura del sistema operativo y se facilita el desarrollo del software en espacio de usuario.
- Multitarea en espacio de *kernel*: dentro del módulo se utilizaron varias tareas *kthreads* para evitar bloqueos entre operaciones de escritura de datos al espacio de usuario con la recepción de nuevos datos por SPI.
- Indexado de memoria contigua: debido a que la escritura por DMA no permite escribir en colas circulares, se implementó un sistema de una cola lineal indexada. Esto permite poder leer y escribir sin solapamientos. Cuando está cerca del límite de utilización se reinician los índices.

En los fragmentos de código 3.4, 3.5 y 3.6 se destacan algunas de estas técnicas.

En el diagrama de bloques de la figura 3.4 se muestra la secuencia de pasos implementada en el módulo.

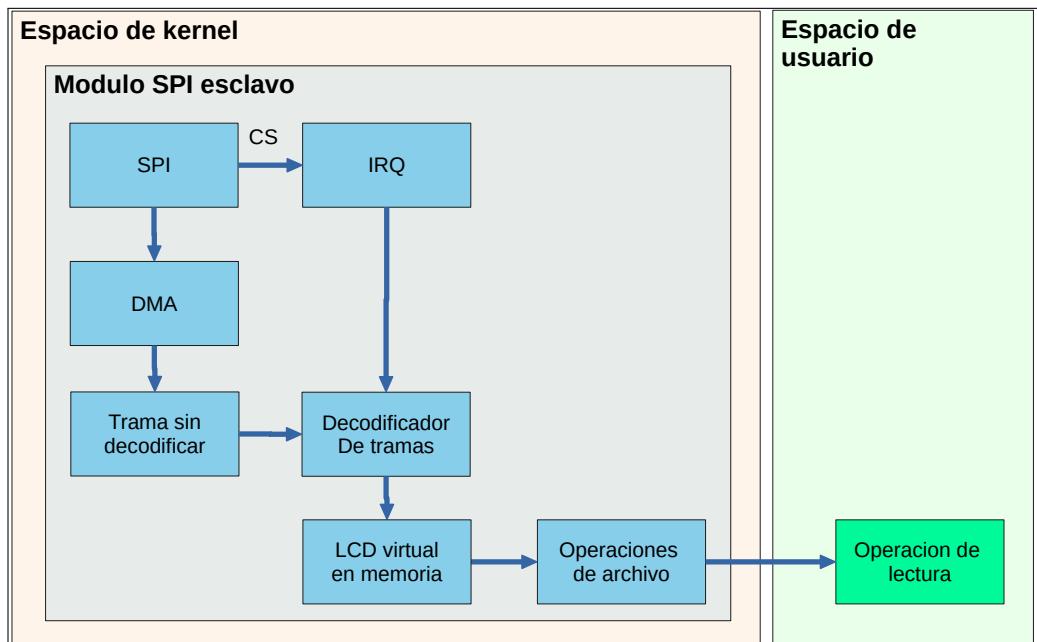


FIGURA 3.4. Diagrama de bloques implementados en el driver de *kernel* para la lectura e interpretación de las tramas de datos entre el controlador y el LCD.

```

1 static irq_handler_t csIrqHandler(unsigned int irq, void *dev_id,
2                                 struct pt_regs *regs)
3 {
4     complete(getNewDataReady());
5     return (irq_handler_t) IRQ_HANDLED;
6 }
```

CÓDIGO 3.4. Manejador de la interrupción asociada a una transacción SPI. Cuando es llamada, activa toda la cadena de eventos que culmina con la actualización del estado de la pantalla.

```

1 static int newDataFunc(void *nol)
2 {
3     int ans;
4     while(1) {
5         ans=wait_for_completion_interruptible_timeout(&newData.ready,
6             msecs_to_jiffies(param_newDataTout));
7         reinit_completion(&newData.ready);
8         newData.actualIndex = findSpiFifoLen(newData.actualIndex);
9         newData.lastIndex = parse ( newData.actualIndex );
10        if(ans==0)
11            wakeUpFileOp();
12    }
13    return 0;
14 }
```

CÓDIGO 3.5. Tarea que espera en modo bloqueante una nueva trama de datos. Utiliza uno de los métodos de comunicación interprocesos del *kernel* de Linux.

3.2.3. Sistema virtual completo

Trabajando en conjunto, el teclado y la pantalla virtual son todo lo necesario para tomar control de la máquina desde la PocketBeagle.

El teclado virtual corre como servicio del sistema operativo y se encarga de la comunicación UART. La pantalla virtual corre como módulo del *kernel* y atiende la interfaz SPI.

En el ejemplo de la figura 3.5 se puede ver en la pantalla LCD el ingreso de los números “1234” sin presionar los botones del teclado físico. Estos son enviados desde una consola de comandos de la PocketBeagle.

Dada la complejidad y la falta de información oficial, el desarrollo y puesta a punto de estos dos subsistemas representaron aproximadamente el 50 % del total del trabajo.

3.2.4. Conexión USB como dispositivo de almacenamiento

La única opción para transferir archivos al controlador es a través un dispositivo de almacenamiento USB o *pendrive*.

```

1 static ssize_t lcd_read( struct file *filp, char __user *buf, size_t
2 count, loff_t *f_pos )
3 {
4     size_t len=0;
5     size_t miss=0;
6     char localBuf[FRAME_LEN];
7
8     int i=(int)filp->private_data;
9     wait_event_interruptible(fileOp.queue[i].ready, fileOp.queue[i].
10 flag);
11     fileOp.queue[i].flag=false;
12     len = ((FRAME_LEN-1)<count)?(FRAME_LEN-1):count;
13     memcpy(localBuf,(char*)getLcd()->cram,LCD_LEN);
14     sprintf(localBuf+LCD_LEN,TRAILER_LEN,"%05u\r\n",fileOp.queue[i].
15 frameNumber++);
16     miss= copy_to_user(buf,localBuf,FRAME_LEN-1);
17     len=len-miss;
18     decWakeUpCounter();
19     return len;
20 };

```

CÓDIGO 3.6. Función que bloquea a la espera de una operación de lectura desde el espacio de usuario. Cuando es llamada, copia la nueva información de la pantalla.



FIGURA 3.5. Control de la máquina mediante la PocketBeagle. Se envían datos a una FIFO y un servicio los direcciona al controlador. Los datos del LCD son capturados por el driver y mostrados en la consola. Se puede ver la sincronía entre el mando a distancia y el virtual.

La PocketBeagle cuenta con una conexión USB cliente que puede actuar como un dispositivo de almacenamiento virtual. Sobre esta tecnología se desarrolló una técnica para intercambiar archivos con el controlador de manera remota a través de Wi-Fi.

En la figura 3.6 se puede ver el conexiónado físico entre la PocketBeagle actuando como USB cliente y el controlador actuando como USB anfitrión o *host*.

La tecnología involucrada en Linux para permitir este funcionamiento es *configFS*[29].

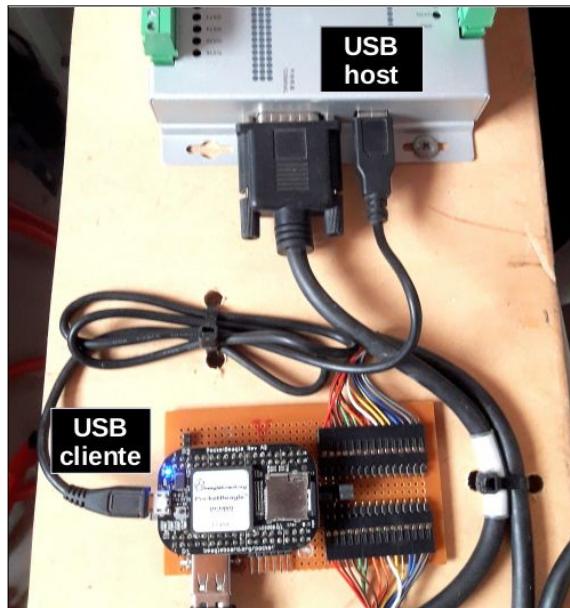


FIGURA 3.6. La PocketBeagle se comporta como un dispositivo de almacenamiento. El controlador accede al sistema de archivos en busca de trabajos a procesar.

Se implementó un sistema de intercambio de doble sistema de archivos como el que se muestra en la figura 3.7. Con este método mientras el controlador accede a un sistema de archivos con trabajos a procesar, la PocketBeagle puede escribir en otro.

Cuando se desea transferir un nuevo archivo al controlador, simplemente se invierten estos dos sistemas: el de lectura pasa a ser escritura y viceversa.

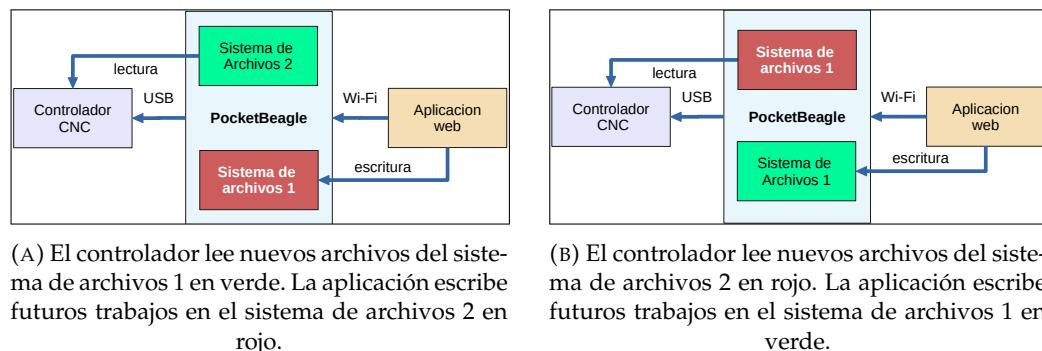


FIGURA 3.7. Técnica de doble sistema de archivos para transferir información al controlador.

En el script mostrado en el listado 3.7 se aprovecha el lenguaje *bash* para implementar este sistema dual en muy pocas líneas y corre como servicio en la PocketBeagle.

3.3. Cámara de vídeo a PocketBeagle

Como se comentó en la sección 2.4, en el alcance de este trabajo se optó por usar la cámara de un teléfono celular montado en la máquina CNC y transmitir vídeo por Wi-Fi con una aplicación.

```

1 pipe=./mass_pipe
2 mnt_point=./mnt_point
3 lun_file=/sys/kernel/config/usb_gadget/g_multi/functions/mass_storage
4 .usb0/lun.0/file
5 fs0=./fs0.img
6 fs1=./fs1.img
7
8 while true
9 do
10   if read line <$pipe; then
11     if test $fs = $fs0
12     then
13       fs=$fs1
14     else
15       fs=$fs0
16     fi
17     mount $fs $mnt_point
18     cp $line $mnt_point
19     umount $mnt_point
20     echo $fs > $lun_file
21   fi
22 done

```

CÓDIGO 3.7. Implementación de doble sistema de archivos conectado con la tecnología configFS. Se aprovecha la potencia de *bash* y se corre como servicio.

En la figura 3.8 se muestra un soporte desarrollado para sujetar el teléfono al eje Z de la máquina.

Además se agregó un soporte para un puntero láser que sirve para ajustar las posiciones relativas del teléfono y el motor de corte.

El dispositivo de sujeción permite ajustar la altura y rotación de manera precisa.

Para recibir las tramas de vídeo desde el celular y procesarlas en la PocketBeagle, se implementó una serie de funciones en lenguaje Python sobre la base de la biblioteca PyOpenCV¹

En el fragmento de código 3.8 se lista el procedimiento para acceder a la cámara y tomar un fotograma de vídeo.

Una vez capturado el fotograma se lo procesa con una secuencia de operaciones que se muestran en el fragmento de código 3.9.

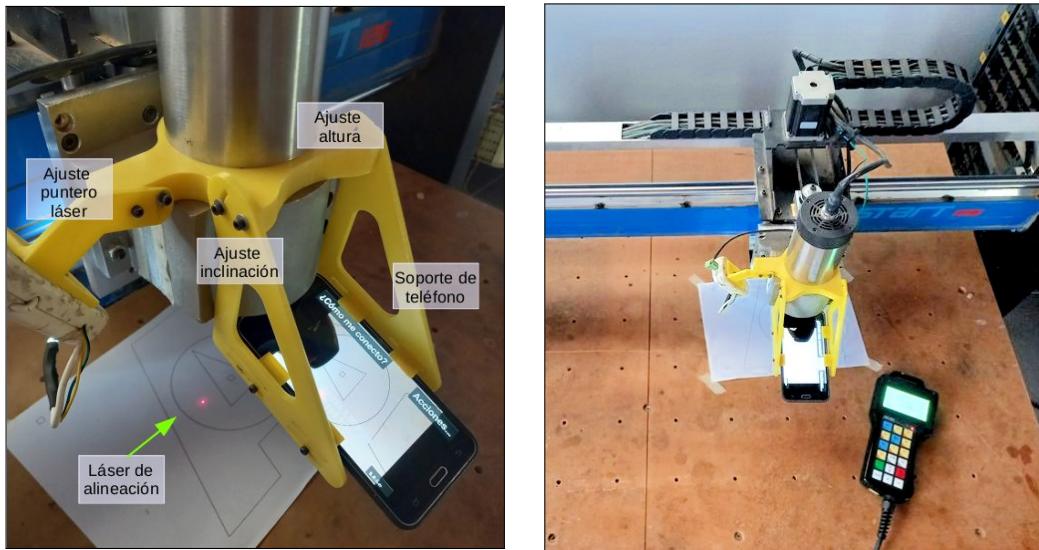
El fotograma “img” está formado por tres canales: azul, verde y rojo o BGR *blue, green, red*. A su vez cada pixel de cada canal está representado por 8 bits de datos.

En la línea 2 del algoritmo se convierte el formato de “img” a escala de grises según la ecuación 3.1:

$$\text{RGB}[A] \text{ to Gray: } Y \leftarrow 0,299 \cdot R + 0,587 \cdot G + 0,114 \cdot B \quad (3.1)$$

Luego en la línea 3 se convierte el formato de escala de grises a binaria de 8 bits, en donde el criterio de selección por 0 o 255 lo decide la variable “grayThresh”.

¹<https://pypi.org/project/pyopencv/>



(A) Soporte de teléfono implementado, con posibilidad de ajustar la altura, rotación e inclinación

(B) Vista superior de la máquina CNC con el soporte del celular y láser instalados.

FIGURA 3.8. Soporte de celular y puntero láser solidarios al eje Z.

```

1 def capture():
2     vcap      = cv.VideoCapture("192.168.1.30:/video", cv.CAP_FFMPEG);
3     ret       = vcap.grab()
4     ret,frame = vcap.retrieve()
5     vcap.release()
6     return frame
7

```

CÓDIGO 3.8. Conexión a la cámara del teléfono por Wi-Fi y captura de un fotograma para su posterior procesamiento.

Un ejemplo de esta secuencia de procesamiento se puede ver en la figura 3.9.

La imagen binaria es procesada en la línea 4 por la función “findContours”² que busca todos los posibles contornos.

Sin embargo se configura para que devuelva solo aquellos que no están encerrados por un contorno mayor.

En el ejemplo de la figura 3.10b se puede ver este efecto en los cuadrados sin relleno que son detectados solo por su contorno externo.

En la siguiente parte del algoritmo se recorren todos los contornos encontrados y se le aplica una función que aproxima cada contorno a uno más suave.

Este algoritmo pretende filtrar irregularidades de la marca o de la captura de vídeo y se configura para que devuelva siempre un perímetro cerrado.

Se utiliza la función minAreaRect en la línea 15 para encontrar un rectángulo de área mínima que encierre completamente al perímetro. Esto permite calcular una buena aproximación del área y del ángulo.

²https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

```

1 def recognition(self,img):
2     imggray           = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
3     ret, thresh       = cv.threshold(imggray, int(self.grayThresh), 0
4     xff, cv.THRESH_BINARY_INV)
5     contours, hierarchy = cv.findContours(thresh, cv.RETR_EXTERNAL, cv
6     .CHAIN_APPROX_SIMPLE)
7
8     validContours      = 0
9     maxArea            = 0
10    maxAreaIndex       = 0
11    maxHull            = 0
12    maxRect            = 0
13    height,width,channels = img.shape
14
15    for i in range ( min(len(contours),10)):
16        hull = cv.approxPolyDP(contours[i],0,True)
17        rect = cv.minAreaRect ( hull )
18        area = rect[1][0]*rect[1][1]
19        if ( area<targetArea*1.5 and area>targetArea*0.5 ) :
20            if ( area>=maxArea ):
21                maxArea      = area;
22                maxAreaIndex = i;
23                maxHull      = hull
24                maxRect      = rect
25                center       = maxRect[0]
26                self.angle   = maxRect[2]

```

CÓDIGO 3.9. Algoritmo principal de reconocimiento de marcas en un fotograma

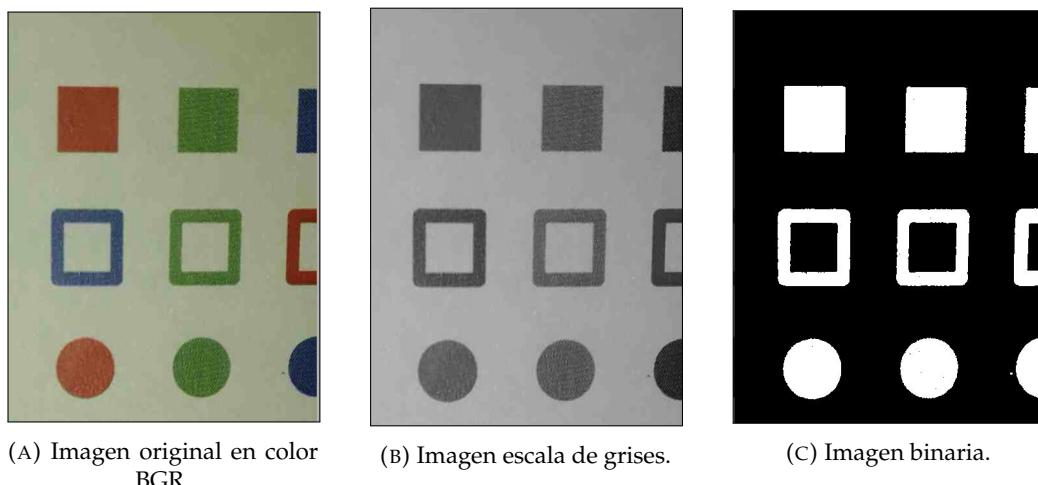


FIGURA 3.9. Imagen original, escala de grises y binaria obtenidas mediante un algoritmo en Python como parte del proceso de reconocimiento de marcas.

Finalmente se selecciona solamente la marca que tenga una área cercana a la especificada por el parámetro “targetArea” y le permite al usuario discriminar una marca por sobre otras.

A la marca resultante se le calcula el desplazamiento en X e Y y su rotación respecto a la cámara.

Estos son los datos que se ingresan en las ecuaciones trigonométricas vistas en la sección 2.5 y es lo que permite realizar la alineación de la pieza.

La secuencia de pasos mencionada se puede ver en la figura 3.10.

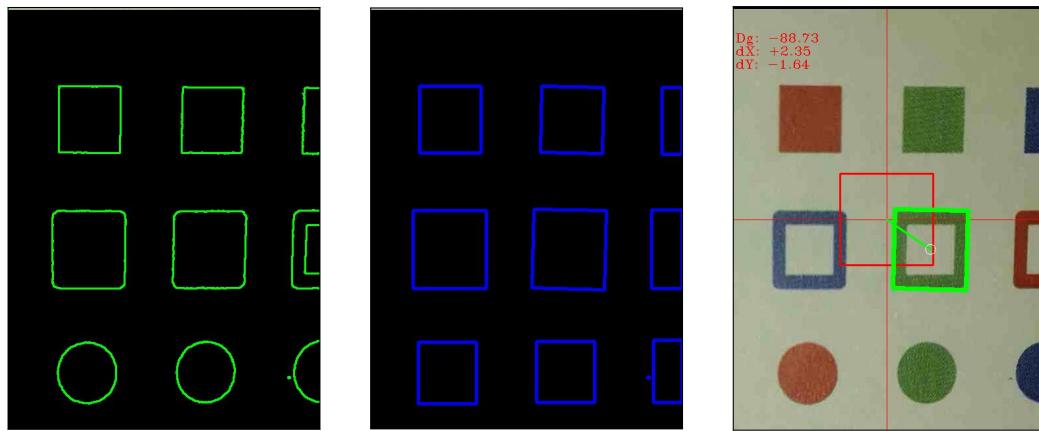


FIGURA 3.10. Proceso de filtrado y selección de perímetros dentro de la imagen. Como resultado se obtiene el centro y rotación de la marca con el área especificada.

En el apéndice A se presentan más ejemplos de capturas de marcas más complejas en donde se visualizan todos los pasos del algoritmo.

3.4. Software de control

El software de control expone los servicios de teclado y pantalla virtual desarrollados, en una página web que permite el control total de la máquina.

Se utilizaron las tecnologías descritas en la sección 2.1 y se pueden ver algunas capturas en la figura 3.11.

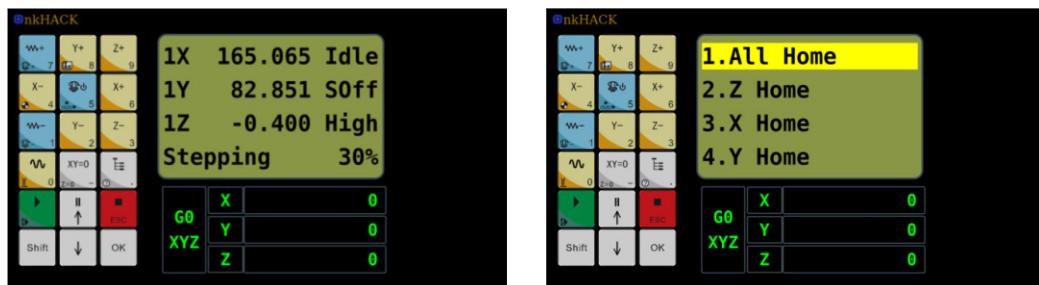


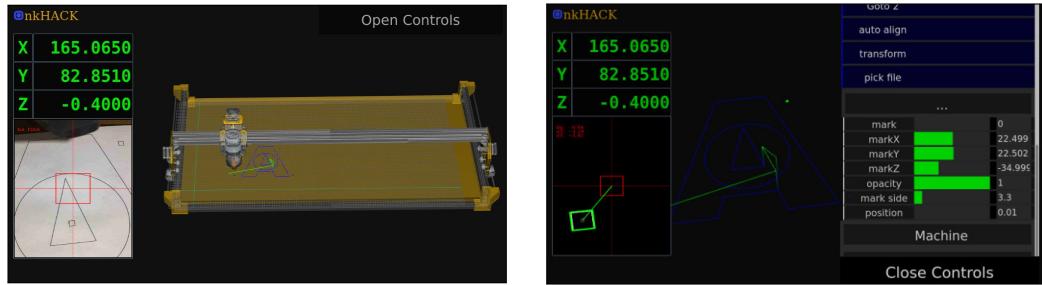
FIGURA 3.11. Página de control de la máquina a través de una página web.

A diferencia del mando a distancia cableado, el control a través de esta página web se puede hacer desde cualquier dispositivo con acceso a la red a la cual pertenece la PocketBeagle.

Por otra parte se agrego una funcionalidad extra que el dispositivo original no cuenta, que es poder enviar la maquina a las coordenadas precisas que se ingresen. Esta funcion es un ejemplo de las posibilidades que adiciona el accesorio.

3.5. Software de lectura de marcas nkHACK

Sobre la base del desarrollo de la pagina web de control, se extendio la aplicacion y se implemento el sistema de alineacion por lectura de marcas. En la figura 3.12 se puede ver la presentacion general de esta nueva pagina.



(A) Pantalla de visualizacion y posicion de la maquina simulada, las coordenadas actuales, la camara y el trabajo en 3D a mecanizar.

(B) Opciones de configuracion de la camara. Se puede ver ahora la camara con imagen binaria detectando una marca de 3.3mm de lado.

FIGURA 3.12. Pagina de control y reconocimiento de marcas.

Algunas de las posibilidades en esta pagina se detallan en la siguiente lista:

- Carga del modelo en 3D de la maquina directamente desde el programa de CAD. Por el momento se soporta el programa Rhinoceros, pero se puede extender a otros formatos.
- Moviento en tiempo real de la maquina simulada segun el recorrido del trabajo.
- Carga de los archivos de trabajo GCode para visualizar y alinear.
- Marcas de registro leidas directamente desde el archivo GCode, ingresada manualmente, o una combinacion de ambas.
- Posibilidad de desplazar, rotar y escalar manualmente un trazo de corte.
- Posibilidad de desplazar, rotar y escalar manualmente un trazo de corte.
- Escala del pixel de la camara en X e Y independientemente.
- Ingreso de la dimension del tamano de la marca.
- Ingreso de la direccion web de la fuente de video.
- Intercambio entre video original a color o imagen binaria.
- Calibracion del nivel de decision para el filtrado de escala de grises a imagen binaria.
- Zoom digital de la imagen capturada en tiempo real para segmentar el area de analisis.
- Transferencia del archivo de corte al controlador.

- Ejecucion del archivo de corte cargado en el controlador con un boton.
- Alineacion del trabajo de corte en modo manual, marca por marca.
- Alineacion del trabajo de corte automaticamente con un solo boton, alinea, transfiere y corta.
- Customizacion completa de la interfaz, permitiendo setear transparencias a cada una de la partes, menus, camara, coordenadas, trazos, maquina, etc.
- Grilla de trabajo customizable.
- Acceso a la pagina de control sin cerrar esta pagina.

3.6. Calibracion de la camara

Para utilizar un fotograma de video como medida para el desplazamiento de una marca, es necesario conocer la distancia que representa cada pixel. Para obtenerlo se aplican las ecuaciones lineales 3.2:

$$\begin{aligned} pixel_x &= \frac{dim_x}{pixeles_x} \\ pixel_y &= \frac{dim_y}{pixeles_y} \end{aligned} \quad (3.2)$$

donde:

- $pixel_x, pixel_y$: dimension en mm de cada pixel en direccion x e y.
- $pixeles_x, pixeles_y$: numero de pixeles de cada fotograma en direccion x e y.
- dim_x, dim_y : medida en mm del fotograma en direccion x e y.

Se ingresan las dimensiones en milimetros de ancho y alto de un cuadro con la ayuda de una regla, y el software realiza el resto de los calculos.

Sin embargo si la camara no esta perfectamente perpendicular a la marca, la dimension de los pixeles en la imagen tienen una dimension diferente dependiendo de su posicion.

Por otro lado aparece un segundo efecto de distorcion de la linealidad en funcion del tipo de lente de la camara y la distancia al objetivo.

En la figura 3.13 se puede ver la captura de una regla en una toma perpendicular y otra en angulo y se puede apreciar que en el ultimo caso se distorcionan las medidas.

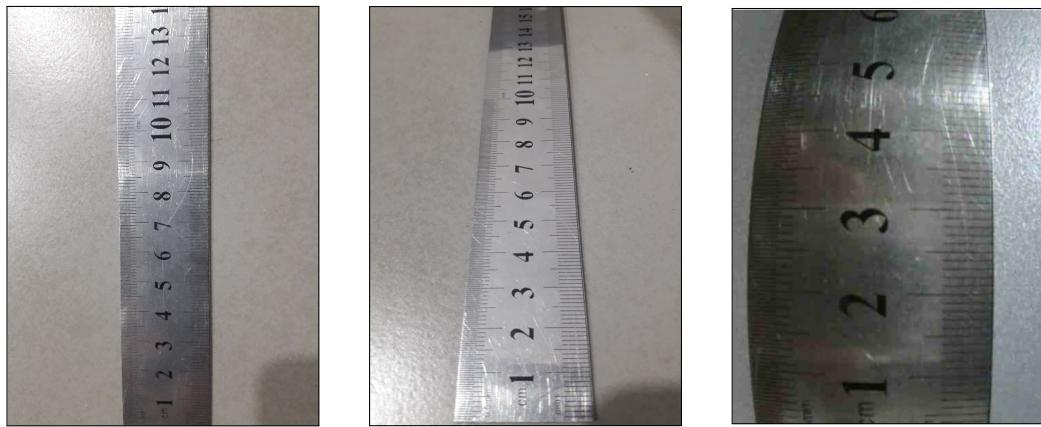
Estas distorcciones afectan la dimension de cada pixel e impactan directamente en el error de alineacion.

Para mitigar este problema se desarrollaron dos tecnicas:

- Zoom: para minimizar el efecto de la curvatura de la imagen en los bordes, se posiciona la camara a cierta distancia tal que permita recortar y descartar los bordes.

En la figura 3.14 se aplica esta correccion a las imagenes de la figura 3.13 y se puede apreciar que en la zona reducida ahora la distorcion es menos apreciable.

- Centrado: se utiliza la dimension del pixel como una aproximacion, pero luego se itera el acercamiento de la maquina hasta que la marca se posiciona en el centro de la imagen con un error parametrizable en el software. Este procedimiento aletarga el proceso de alineacion, pero minimiza los efectos de error mencionados, debido a que la distancia entre el centro de la imagen y la marca tienden a cero pixeles.
- En la figura 3.15 se muestra una marca detectada a cierta distancia de lo esperado y como al aplicar este metodo, se logra luego de varias iteraciones posicionar exactamente la marca en el centro de la imagen en donde los errores de la no linealidad de la imagen son despreciables



(A) Toma perpendicular. Se puede ver cierta linealidad en toda la imagen.

(B) Toma en angulo. Se puede ver la distorcion de las medidas de la regla en funcion de la posicion en la imagen.

(C) Fotografia tomada por una lente de gran angular “ojo de pez” que resalta el efecto de distorcion.

FIGURA 3.13. Revelacion del efecto de distorcion de las medidas en funcion del angulo de captura de la imagen, tipo de lente y distancia al objetivo.



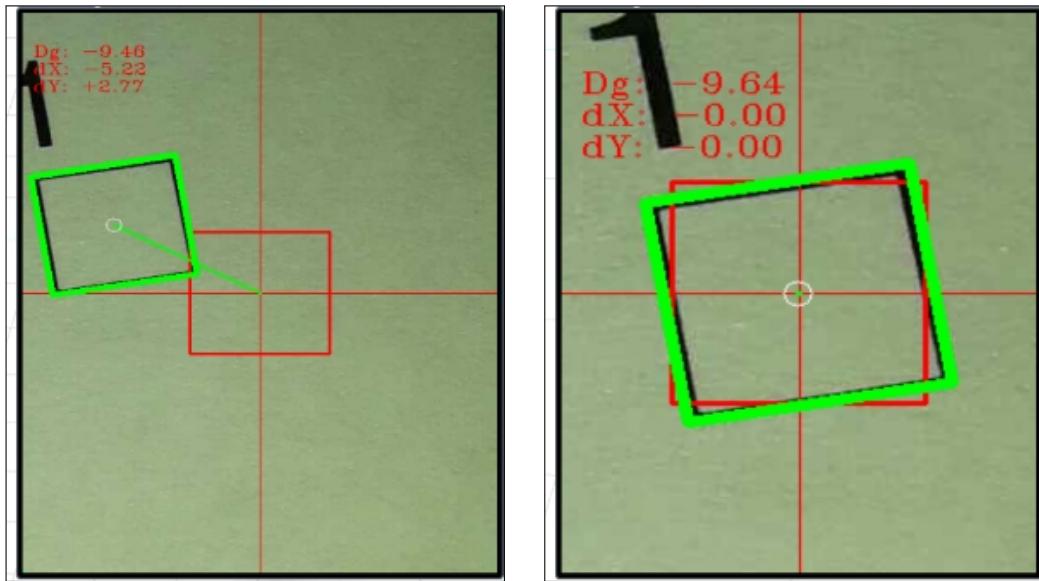
(A) Recorte de imagen perpendicular. Se aprecia cierta linealidad en la imagen.

(B) Recorte de una imagen tomada en angulo. Aun con la distorcion generada, el recorte permite tomar una zona local sensiblemente mas lineal que la imagen completa.

(C) Recorte de una imagen tomada con gran angular. Aun se puede apreciar algo de distorcion.

FIGURA 3.14. Analisis de la distorcion en las medidas de imagenes tomadas en diferentes escenarios y recortadas para tomar solo una zona central de la imagen y reducir la distorcion.

3.7. Secuencia de pasos para alinear



(A) Marca encontrada a cierta distancia del centro de la imagen. El desplazamiento real depende de la medida de cada pixel y de la linealidad de la imagen.

(B) Interacion de movimientos que ubica la marca en el centro de la imagen. El desplazamiento del centro es prácticamente nulo, y por lo tanto no depende de la dimensión de los pixeles ni la linealidad de la imagen.

FIGURA 3.15. Método de iteración de movimientos para el centrado de la marca en el centro de la imagen y la mitigación de los efectos alineales en el fotograma.

TABLA 3.2. Detalle paso a paso de las secuencia necesaria para alinear y cortar un trabajo.

Procedimiento	Imagen
Paso0	

TABLA 3.2. Detalle paso a paso de las secuencia necesaria para alinear y cortar un trabajo. Continuacion.

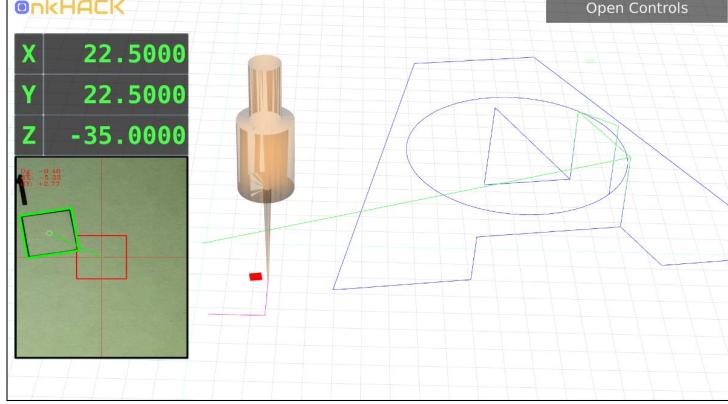
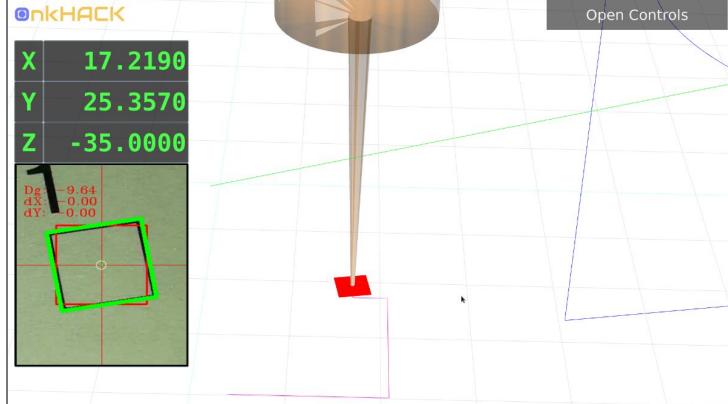
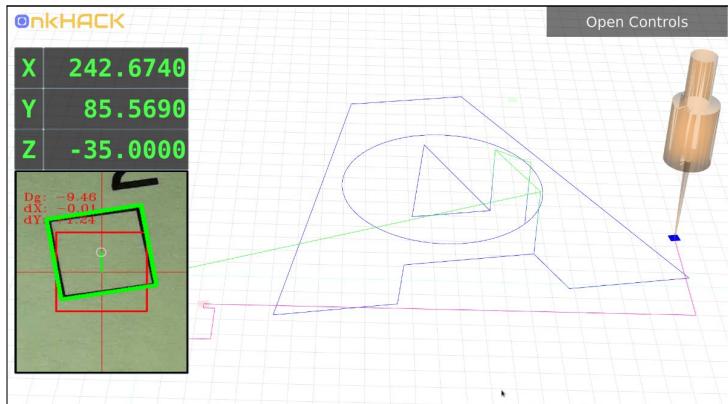
Procedimiento	Imagen
Paso1	
Paso1 aligned	
Paso2	

TABLA 3.2. Detalle paso a paso de las secuencia necesaria para alinear y cortar un trabajo. Continuacion.

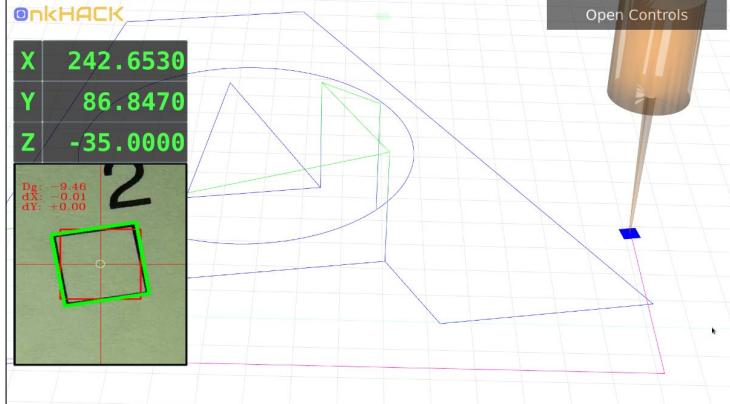
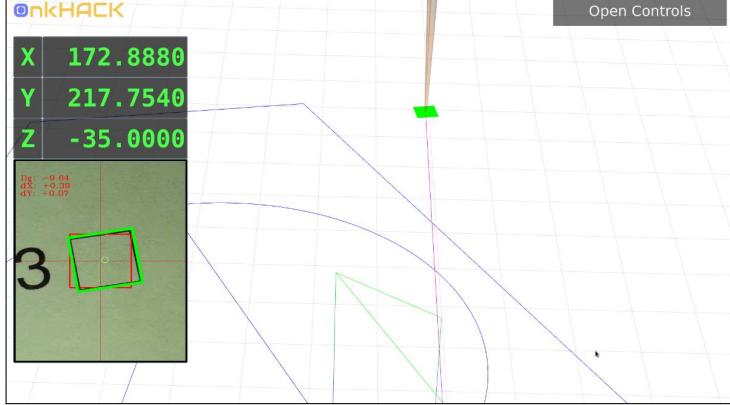
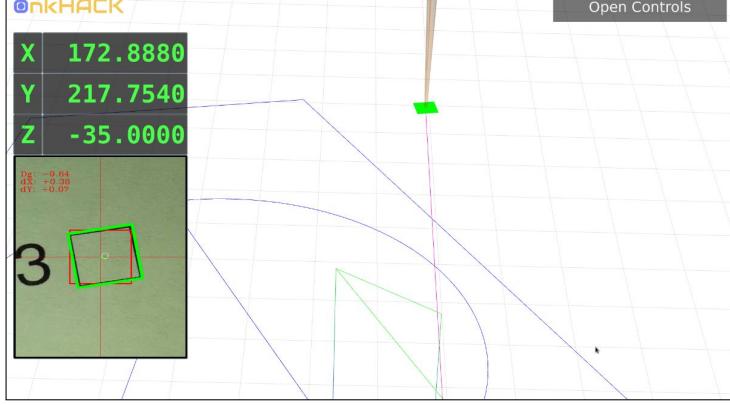
Procedimiento	Imagen
Paso2 aligned	
Paso3	
Paso3 aligned	

TABLA 3.2. Detalle paso a paso de las secuencia necesaria para alinear y cortar un trabajo. Continuacion.

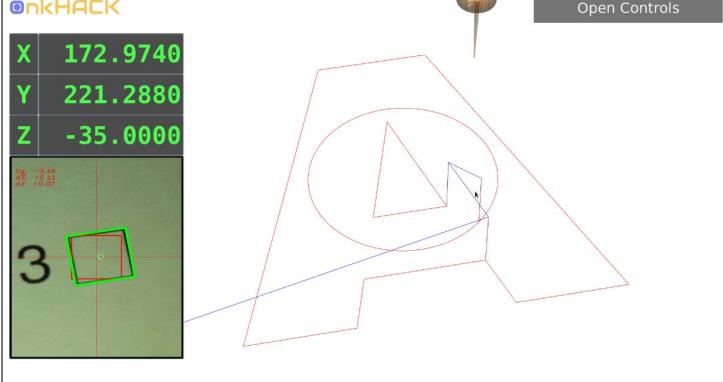
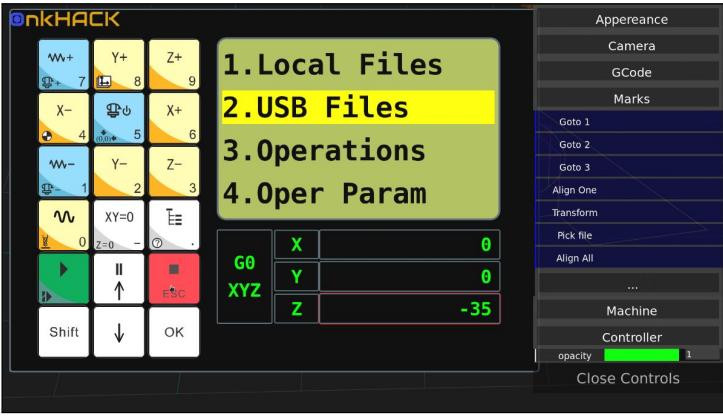
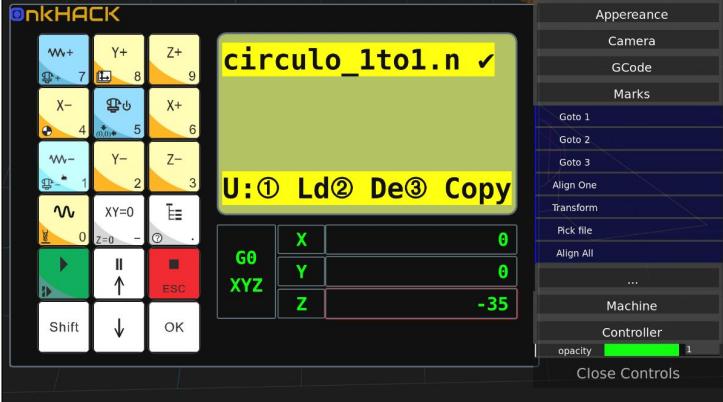
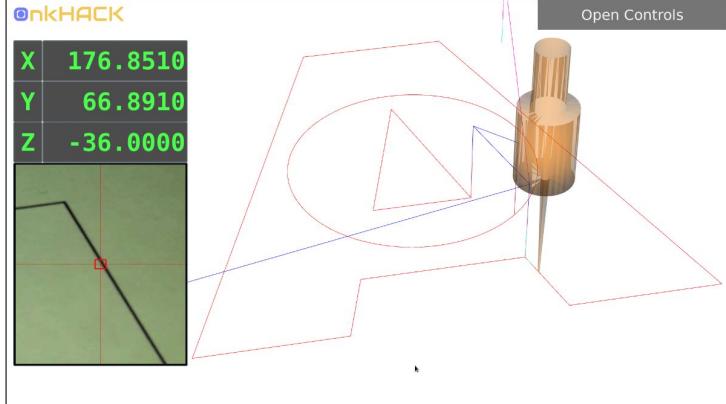
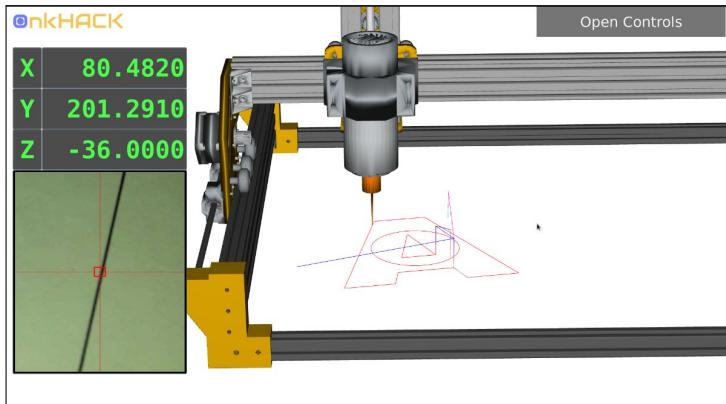
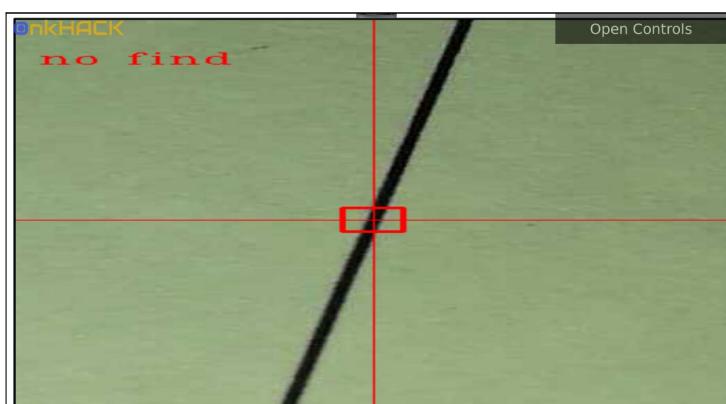
Procedimiento	Imagen
Paso transform	
Paso pick file 1	
Paso pick file 2	

TABLA 3.2. Detalle paso a paso de las secuencia necesaria para alinear y cortar un trabajo. Continuacion.

Procedimiento	Imagen
Paso pick path1	
Paso pick path2	
Paso pick path3	

Apéndice A

Ejemplos del proceso de reconocimiento de marcas

Se exponen algunos ejemplos de reconocimiento de marcas presentados en una secuencia que se detalla en la siguiente lista:

- Paso 1: imagen original a color.
- Paso 2: escala de grises.
- Paso 3: imagen binaria.
- Paso 4: detección de todos los bordes externos.
- Paso 5: cálculo de área mínima rectangular.
- Paso 6: cálculo de ángulo y posición de la marca seleccionada.

Se puede ver en la imagen A.1b que si la imagen es irregular, la cantidad de contornos detectados esta tan grande que afecta el rendimiento del algoritmo de detección y la performance del sistema.

Para mitigar este efecto, en los algoritmos se limita la cantidad de bordes permitidos.

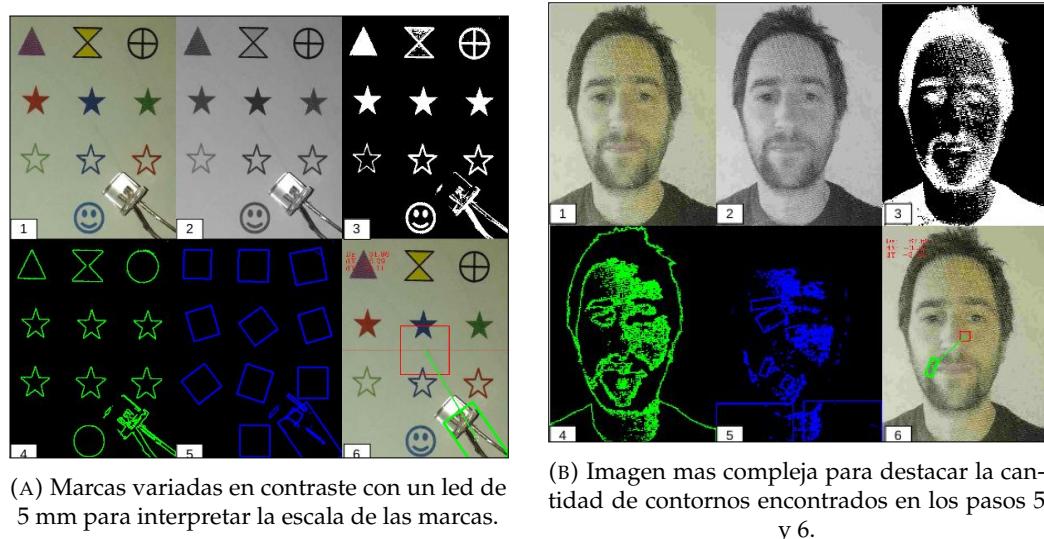


FIGURA A.1. Secuencia de reconocimiento de marcas.

En la figura ?? pasos 4 y 5 se destaca el efecto de un contorno que está formado por la unión de muchas figuras. Sin embargo la detección de área mínima lo toma

como un solo contorno. Esto se debe a que las figuras se están tocando en algunos puntos, y el algoritmo lo considera un solo perímetro irregular.

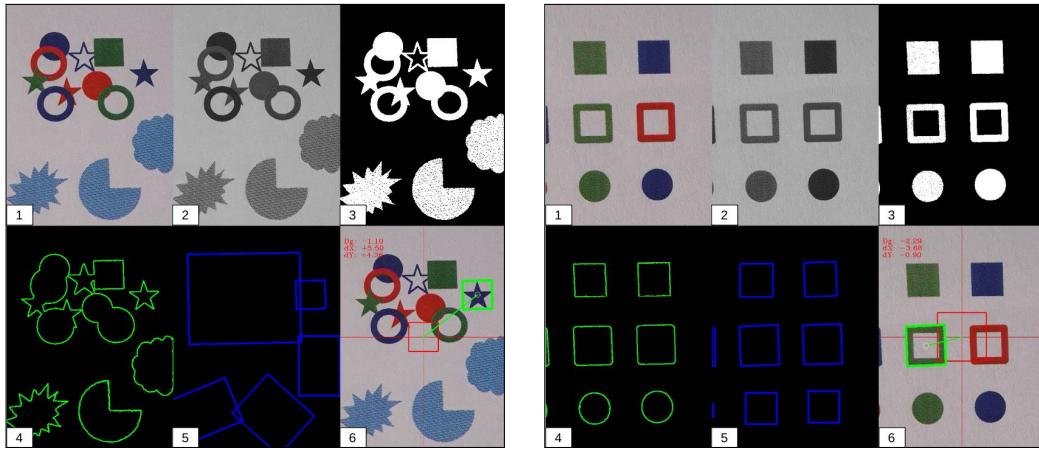


FIGURA A.2. Secuencia de reconocimiento de marcas.

Bibliografía

- [1] wikipedia.com. *La especificacion del lenguaje GCode.*
<https://en.wikipedia.org/wiki/G-code#Implementations>. Ene. de 2019.
 (Visitado 10-11-2020).
- [2] Elena R. Messina Thomas R. Kramer Frederick M. Proctor. *The NIST RS274NGC Interpreter - Version 3.* 6556. Rev. 3. NIST National Institute of Standards y Technology. 2000.
- [3] https://en.wikipedia.org/wiki/Numerical_control.
- [4] <https://www.edingcnc.com/>.
- [5] <http://www.cnccamera.nl/src-gen/products.html>.
- [6] <https://wolfcut.es/>.
- [7] <https://www.pv-automation.com/en/>.
- [8] <https://www.pv-automation.com>.
- [9] <https://www.machinekit.io/>.
- [10] <https://www.summa.com/en/solutions/>.
- [11] <https://www.weihong.com.cn/en/products/controller/20160424/23.html>.
- [12] *Ecosistema de placas de desarrollo de proposito general.*
<https://http://beagleboard.org/>. (Visitado 02-02-2021).
- [13] <https://www.xilinx.com/products/boards-and-kits/1-hydd4z.html>.
- [14] [https://www.python.org/](https://www.python.org).
- [15] <https://docs.python.org/3/library/asyncio.html>.
- [16] <https://docs.aiohttp.org/en/stable/>.
- [17] <https://en.wikipedia.org/wiki/HTML5>.
- [18] <https://en.wikipedia.org/wiki/CSS>.
- [19] <https://en.wikipedia.org/wiki/JavaScript>.
- [20] <https://get.webgl.org/>.
- [21] <https://https://threejs.org/>.
- [22] <https://en.wikipedia.org/wiki/WebSocket>.
- [23] <https://socket.io/>.
- [24] <https://opencv.org/>.
- [25] <https://pypi.org/project/pyopencv/>.
- [26] <https://play.google.com/store/apps/details?id=com.pas.webcam&hl=en&gl=US>.
- [27] Christopher Hallinan. *Embedded Linux Primer, Second Edition.* A Practical, Real-World Approach. 2010.
- [28] Alessandro Rubini by Jonathan Corbet y Greg Kroah-Hartman. *Linux Device Drivers, Third Edition.* 6556. Third Edition. 2005.
- [29] https://www.kernel.org/doc/Documentation/usb/gadget_configs.txt.