

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	6
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	17
2.5. Trigonometría de alineación	17
2.6. Detección y tipos de marcas fiduciales	21
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. Calibración de la cámara	38
3.7. Secuencia de pasos para alinear	40
4. Ensayos y resultados	45
4.1. Listado de herramientas	45
4.2. Pruebas funcionales del hardware	45
4.2.1. Acceso concurrente al driver SPI	45
4.2.2. Ensayo con archivos de mecanizado	46
4.2.3. Ensayos con impresiones escaladas	48
4.2.4. Ensayos con diferentes resoluciones de vídeo	50
5. Conclusiones	53
5.1. Conclusiones generales	53

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	6
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	11
2. Introducción específica	13
2.1. Tecnologías utilizadas	13
2.2. Plataforma PocketBeagle	13
2.3. Aplicación web	14
2.4. Cámara de vídeo	17
2.5. Trigonometría de alineación	17
2.6. Detección y tipos de marcas fiduciales	21
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. Calibración de la cámara	38
3.7. Secuencia de pasos para alinear	40
4. Ensayos y resultados	45
4.1. Listado de herramientas	45
4.2. Pruebas funcionales del hardware	45
4.2.1. Acceso concurrente al driver SPI	45
4.2.2. Ensayo con archivos de mecanizado	46
4.2.3. Ensayos con impresiones escaladas	48
4.2.4. Ensayos con diferentes resoluciones de vídeo	50
5. Conclusiones	53
5.1. Conclusiones generales	53

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.	34
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. Página web de control de la máquina. Ejemplos de operación.	37
3.12. Pantalla principal del software nkHACK. Se visualiza el modelo de la máquina CNC, las coordenadas actuales, la imagen de la cámara y el trabajo a mecanizar.	37
3.13. Software nkHACK. Menú de configuración de la cámara. Se puede ver la imagen capturada en modo binaria en la cual se ha detectado una marca de 3.3mm de lado.	38
3.14. Revelación del efecto de distorsión de las medidas en función del ángulo de captura de la imagen, distancia al objetivo y tipo de lente.	39
3.15. Análisis de la distorsión en las imágenes tomadas en diferentes escenarios. Se aplican técnicas de recorte para mitigar el defecto.	40
3.16. Método de iteración de movimientos para el centrado de la marca y la mitigación de los efectos alineales en el fotograma.	41
 4.1. Acceso concurrente al módulo de manejo de SPI desde una conexión SSH. Se puede ver la sincronía entre los cuatro accesos y el registro de cada acceso en los mensajes del <i>kernel</i>	46
4.2. Placa de madera con la impresión del trabajo de corte pegada. Esto permite ubicar la pieza en diferentes posiciones y probar los resultados del sistema de alineación.	47
4.3. Secuencia de pasos para los ensayos de corte simulado. Se utiliza el recuadro rojo en el centro de la imagen como testigo del error máximo.	47
4.4. Generación de un trabajo de corte perimetral y otro escalado con el objetivo de validar la función de escalado no lineal del software.	48
4.5. Secuencia de pasos para la simulación de corte perimetral con su respectivo archivo de corte GCode sin distorsión.	49
4.6. Secuencia de pasos para la simulación de corte perimetral escalado con el archivo de corte GCode de la versión sin escalar. Se puede notar que las marcas dos y tres aparecen distantes del lugar esperado debido al escalado.	49
A.1. Secuencia de reconocimiento de marcas.	57
A.2. Secuencia de reconocimiento de marcas.	58

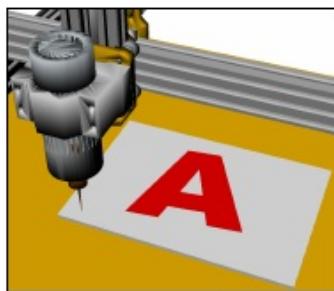
3.5. Control de la máquina mediante la PocketBeagle. Se envían datos a una FIFO y un servicio los direcciona al controlador. Los datos del LCD son capturados por el driver y mostrados en la consola. Se puede ver la sincronía entre el mando a distancia y el virtual.	31
3.6. La PocketBeagle se comporta 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.	34
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. Página web de control de la máquina. Ejemplos de operación.	37
3.12. Pantalla principal del software nkHACK. Se visualiza el modelo de la máquina CNC, las coordenadas actuales, la imagen de la cámara y el trabajo a mecanizar.	37
3.13. Software nkHACK. Menú de configuración de la cámara. Se puede ver la imagen capturada en modo binario en la cual se ha detectado una marca de 3.3mm de lado.	38
3.14. Revelación del efecto de distorsión de las medidas en función del ángulo de captura de la imagen, distancia al objetivo y tipo de lente.	39
3.15. Análisis de la distorsión en las imágenes tomadas en diferentes escenarios. Se aplican técnicas de recorte para mitigar el defecto.	40
3.16. Método de iteración de movimientos para el centrado de la marca y la mitigación de los efectos alineales en el fotograma.	41
 4.1. Acceso concurrente al módulo de manejo de SPI desde una conexión SSH. Se puede ver la sincronía entre los cuatro accesos y el registro de cada acceso en los mensajes del <i>kernel</i>	46
4.2. Placa de madera con la impresión del trabajo de corte pegada. Esto permite ubicar la pieza en diferentes posiciones y probar los resultados del sistema de alineación.	47
4.3. Secuencia de pasos para los ensayos de corte simulado. Se utiliza el recuadro rojo en el centro de la imagen como testigo del error máximo.	47
4.4. Generación de un trabajo de corte perimetral y otro escalado con el objetivo de validar la función de escalado no lineal del software.	48
4.5. Secuencia de pasos para la simulación de corte perimetral con su respectivo archivo de corte GCode sin distorsión.	49
4.6. Secuencia de pasos para la simulación de corte perimetral escalado con el archivo de corte GCode de la versión sin escalar. Se puede notar que las marcas dos y tres aparecen distantes del lugar esperado debido al escalado.	49
A.1. Secuencia de reconocimiento de marcas.	57
A.2. Secuencia de reconocimiento de marcas.	58

Índice de tablas

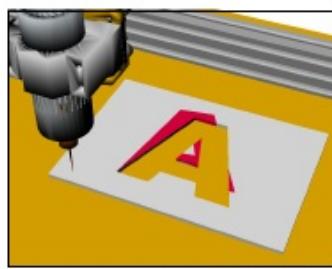
1.1. Modelos de controladores	5
1.2. Modelos de drivers	6
1.3. Sistemas de reconocimiento de marcas	11
2.1. Seleccion de la cámara	17
3.1. Características de los bloques principales	24
3.1. Características de los bloques principales. Continuación	25
3.2. Secuencia de pasos para alinear un trabajo	41
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
4.1. Ensayos de corte simulado	47
4.2. Ensayos de corte simulado escalado	49
4.3. Ensayos de resolucion de imagen	50
4.3. Ensayos de resolucion de imagen	51

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. Seleccion 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	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
4.1. Ensayos de corte simulado	47
4.2. Ensayos de corte simulado escalado	49
4.3. Ensayos de resolucion de imagen	50
4.3. Ensayos de resolucion de imagen	51
4.3. Ensayos de resolucion de imagen	52



(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.

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

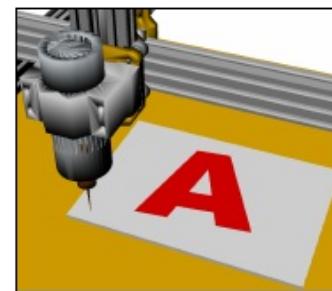
- Alineación de placas de circuito impreso de dos caras.
- Necesidad de volver a alinear una pieza que requiere un nuevo proceso de mecanizado.
- Necesidad de volver a alinear luego de abortar un mecanizado ante un corte de energía.
- Errores de escala y escuadra entre las diferentes máquinas involucradas 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.

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. Mediante el uso de una cámara de video 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.

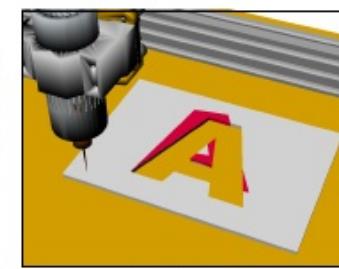
1.3. Software para el reconocimiento de marcas

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 y/o extensiones de software, el costo de las cámaras de video requeridas en estos sistemas son muy costosas.



(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.

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

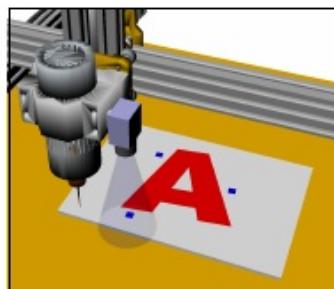
- Alineación de placas de circuito impreso de dos caras.
- Necesidad de volver a alinear una pieza que requiere un nuevo proceso de mecanizado.
- Necesidad de volver a alinear luego de abortar un mecanizado ante un corte de energía.
- Errores de escala y escuadra entre las diferentes máquinas involucradas 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.

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. Mediante el uso de una cámara de video 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.

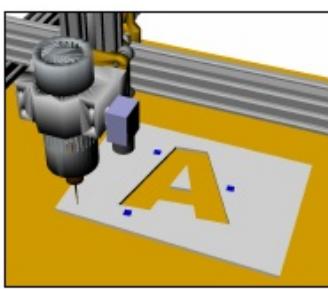
1.3. Software para el reconocimiento de marcas

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. Esto aumenta el costo general del sistema, disminuye la fiabilidad y limita el acceso desde múltiples dispositivos y plataformas.



(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.

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 que aproveche el uso de la cámara de un teléfono celular.

Tampoco se ha encontrado una 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.

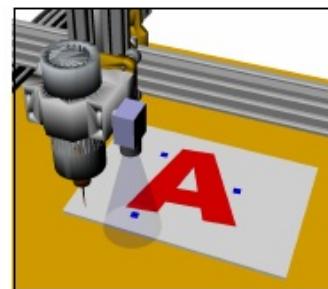
Sin embargo las soluciones que ha utilizado hasta el momento requieren el uso de una PC con Windows para su funcionamiento y la empresa considera que estas tecnologías no son adecuadas para el ámbito industrial de sus clientes.

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

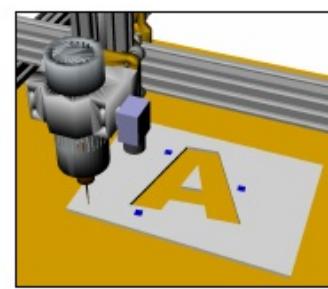
1.5. Motivación y alcance

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.

²<https://wolfcut.es/>



(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.

Además del hardware, las licencias y extensiones de software, las cámaras de video compatibles son muy específicas y de muy alto valor, agravando aún más la viabilidad de estas soluciones.

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 que aproveche el uso de la cámara de un teléfono celular.

Tampoco se ha encontrado una 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.

²<https://wolfcut.es/>

1.5. Motivación y alcance

11

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 y Wolfcut lo han extendido para soportar reconocimiento de marcas.	
myCNC [5]: esta aplicación de la empresa pv-automation [6] ofrece un sistema de visión artificial y reconocimiento de marcas basado en una PC industrial y cámaras USB.	
Machinekit [7]: es un software de control que opera sobre Linux al cual se le han hecho algunas intervenciones para el reconocimiento de marcas.	
Summa [8]: es una línea de máquinas de corte de contornos, principalmente en papel, que cuenta con lectura de marcas integrada en sus sistemas embebidos.	

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 [9] que se muestra en la figura 1.11.

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

1.5. Motivación y alcance

11

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 integrada en sus sistemas embebidos.	

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



FIGURA 1.11. Controlador NK105 de la firma Weihong.

No provee una API (*application programming interface*) definida por el fabricante ni una canal físico para conectarse y extender sus funciones.

Por otro lado, gracias a su diseño basado en FPGA (*field programmable gate array*) y no depender de una PC para funcionar es reconocido por: excelentes resultados de corte, gran estabilidad en trabajos extensos, muy buena 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.

Además este modelo pertenece a una familia de soluciones de complejidad creciente, pero que comparten el controlador. Las extensiones que se incorporen en este se pueden aplicar 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.

la firma Weihong [11] que se muestra en la figura 1.11.



FIGURA 1.11. Controlador NK105 de la firma Weihong.

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

No provee una API (*application programming interface*) definida por el fabricante ni una canal físico para conectarse y extender sus funciones.

Por otro lado, gracias a su diseño basado en FPGA (*field programmable gate array*) y no depender de una PC para funcionar es reconocido por: excelentes resultados de corte, gran estabilidad en trabajos extensos, muy buena 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.

Además este modelo pertenece a una familia de soluciones de complejidad creciente, pero que comparten el controlador. Las extensiones que se incorporen en este se pueden aplicar 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.

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 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.

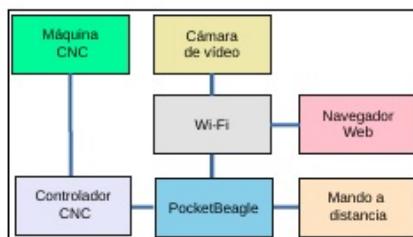


FIGURA 2.1. Diagrama de bloques del sistema implementado.

2.2. Plataforma PocketBeagle

PocketBeagle es un miembro del ecosistema de plataformas de desarrollo BeagleBoard. [10]. 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.

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 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.

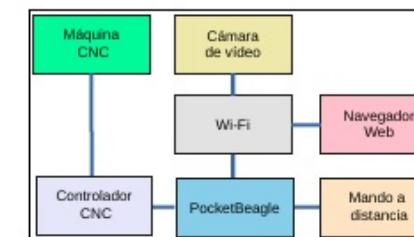


FIGURA 2.1. Diagrama de bloques del sistema implementado.

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:

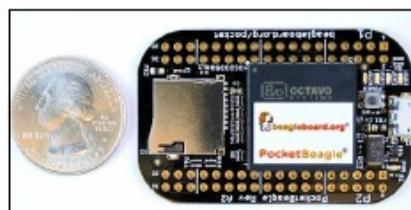
- Controlador integrado SiP (system-in-package) Octavo Systems OSD3358-SM.
- Memoria de 512MB DDR3.
- Unidad de procesamiento de 32b Cortex-A8 @1-GHz.
- 72 pines de expansión, UART, SPI, I2C, entre otras.
- USB de alta velocidad.

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 que se puede descargar desde el link oficial².



(A) Plataforma PocketBeagle como unidad de procesamiento.



(B) Adaptador USB a Wi-Fi que otorga conectividad.

FIGURA 2.2. Conjunto de herramientas adoptadas.

Se evaluaron plataformas más potentes como la PYNQ-Z1 [11]. 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>

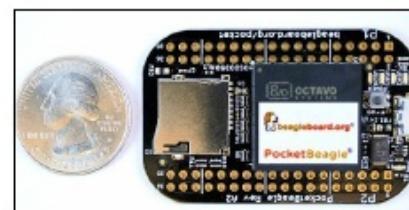
²<https://beagleboard.org/latest-images>

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 que se puede descargar desde el link oficial².



(A) Plataforma PocketBeagle como unidad de procesamiento.



(B) Adaptador USB a Wi-Fi que otorga conectividad.

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>

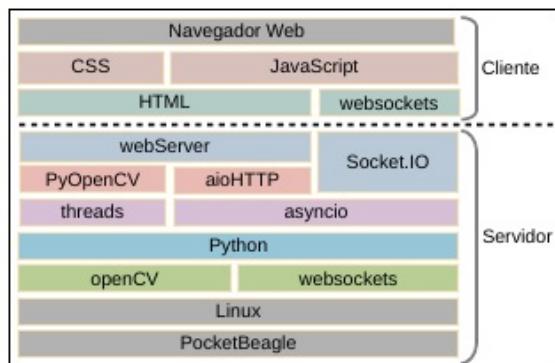


FIGURA 2.3. Capas de software relacionadas con la aplicación web utilizadas.

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

- **Python [12]:** 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 [13]:** 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 [14]:** 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.

- **HTML 5.0 [15]:** 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 [16]:** 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 [17]:** Es un lenguaje de programación intrínsecamente relacionado con HTML que permite la creación de páginas web dinámicas.

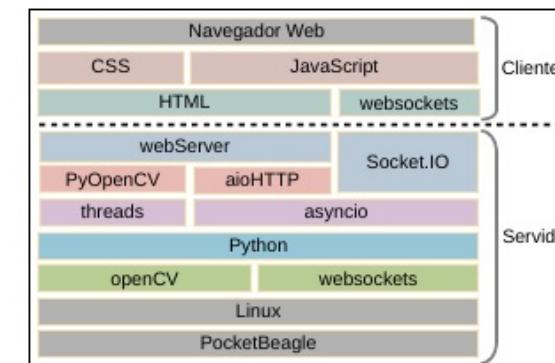


FIGURA 2.3. Capas de software relacionadas con la aplicación web utilizadas.

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.

- **HTML 5.0 [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 [18]: Es una biblioteca gráfica (*Web graphics library*) escrita en JavaScript, que permite definir y visualizar objetos en tres dimensiones en una página web.

Esta intimamente 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 [19]: 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 [20]: 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 [21]: 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 [22]: 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 [23]: 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 que internamente utiliza libopencv-dev del sistema operativo.

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 intimamente 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 que internamente utiliza libopencv-dev del sistema operativo.

2.4. Cámara de video

Los criterios para la selección de la cámara de video 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.

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

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 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 [24] 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 donde se pueden ajustar los parámetros de la cámara más importantes: zoom, brillo, desplazamiento, resolución y calidad de imagen.

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.

2.4. Cámara de video

Los criterios para la selección de la cámara de video 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.

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

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 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 donde se pueden ajustar los parámetros de la cámara más importantes: zoom, brillo, desplazamiento, resolución y calidad de imagen.

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.

3.2. Bloque PocketBeagle

27

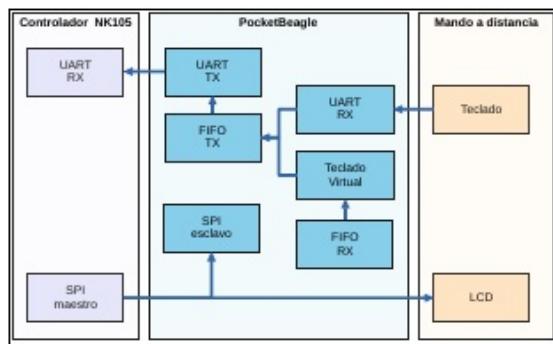


FIGURA 3.3. Diagrama de bloques del software de virtualización de teclado y pantalla.

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]!=
6 FRAME_HEADER)
7             ;
8         PollComport(PORT_NUMBER, frame+1, FRAME_SIZE-1);
9         if(memcmp(frame,FRAME_DEFAULT,FRAME_SIZE))
10             mq_send(msgQueue,frame,FRAME_SIZE,1);
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 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 [25].

3.2. Bloque PocketBeagle

27

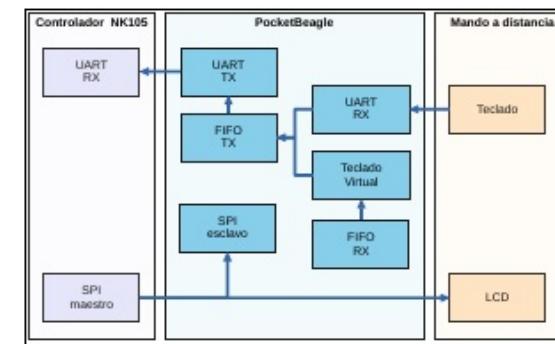


FIGURA 3.3. Diagrama de bloques del software de virtualización de teclado y pantalla.

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]!=
6 FRAME_HEADER)
7             ;
8         PollComport(PORT_NUMBER, frame+1, FRAME_SIZE-1);
9         if(memcmp(frame,FRAME_DEFAULT,FRAME_SIZE))
10             mq_send(msgQueue,frame,FRAME_SIZE,1);
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 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* rcvVirtual(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 }

```

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 }

```

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 [26]. 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:

```

1 void* rcvVirtual(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 }

```

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 }

```

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 estudiaron 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:

3.2. Bloque PocketBeagle

29

- 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 [26].
- 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 *khreads* 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.

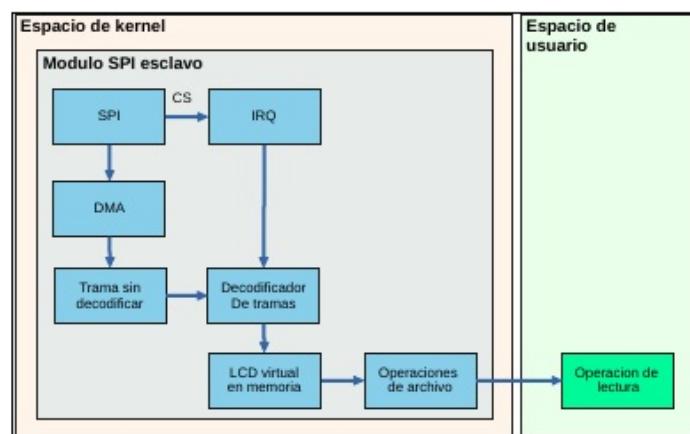


FIGURA 3.4. Diagrama de bloques implementados en el driver de *kernel* para la lectura e interpretación de las tramas de datos entre el controlador y el LCD.

3.2. Bloque PocketBeagle

29

- 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 *khreads* 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.

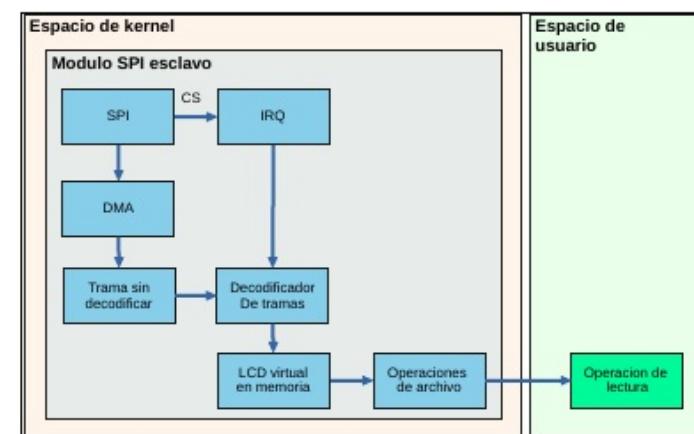


FIGURA 3.4. Diagrama de bloques implementados en el driver de *kernel* para la lectura e interpretación de las tramas de datos entre el controlador y el LCD.

3.2. Bloque PocketBeagle

31

```

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

```

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 envian 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 conexionado 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](#)[27].

3.2. Bloque PocketBeagle

31

```

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

```

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 envian 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 conexionado 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].

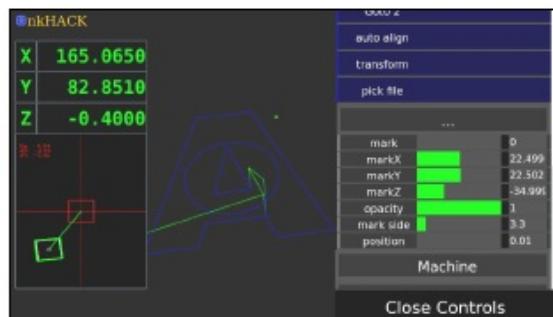


FIGURA 3.13. Software nkHACK. Menú de configuración de la cámara. Se puede ver la imagen capturada en modo **binaria** en la cuál se ha detectado una marca de **3.3mm** de lado.

- Ingreso de la dirección web de la cámara de video.
- Intercambio entre video a color o imagen binaria.
- Calibración del nivel de sensibilidad de la imagen binaria.
- Zoom digital de la imagen capturada en tiempo real para segmentar el área de análisis.
- Transferencia del archivo de corte al controlador.
- Ejecución del archivo de corte cargado en el controlador.
- Alineación del trabajo de corte en modo manual o automático.
- Un solo botón asignado para las tareas de: corregir la alineación según las marcas, transferir el archivo y solicitar el corte.
- Opciones de ajuste de la interfaz visual que permiten ocultar: menús, imagen de la cámara, coordenadas, trazos, modelo de la máquina, etc.
- Grilla de la mesa de trabajo ajustable.
- Acceso a la página de control manual del dispositivo sin cerrar la simulación.

3.6. Calibración de la cámara

Para extraer de un fotograma de vídeo el desplazamiento de una marca al centro de la imagen, es necesario conocer la distancia que representa cada píxel.

Para obtenerlo se aplican las ecuaciones lineales 3.2:

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

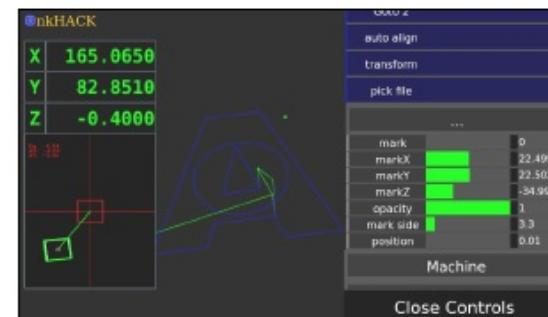


FIGURA 3.13. Software nkHACK. Menú de configuración de la cámara. Se puede ver la imagen capturada en modo **binaria** en la cuál se ha detectado una marca de **3.3mm** de lado.

- Ingreso de la dirección web de la cámara de video.
- Intercambio entre video a color o imagen binaria.
- Calibración del nivel de sensibilidad de la imagen binaria.
- Zoom digital de la imagen capturada en tiempo real para segmentar el área de análisis.
- Transferencia del archivo de corte al controlador.
- Ejecución del archivo de corte cargado en el controlador.
- Alineación del trabajo de corte en modo manual o automático.
- Un solo botón asignado para las tareas de: corregir la alineación según las marcas, transferir el archivo y solicitar el corte.
- Opciones de ajuste de la interfaz visual que permiten ocultar: menús, imagen de la cámara, coordenadas, trazos, modelo de la máquina, etc.
- Grilla de la mesa de trabajo ajustable.
- Acceso a la página de control manual del dispositivo sin cerrar la simulación.

3.6. Calibración de la cámara

Para extraer de un fotograma de vídeo el desplazamiento de una marca al centro de la imagen, es necesario conocer la distancia que representa cada píxel.

Para obtenerlo se aplican las ecuaciones lineales 3.2:

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

donde:

- $pixel_x, pixel_y$: dimensión en mm de cada píxel en dirección X e Y.
- $pixels_x, pixels_y$: número de píxeles de cada fotograma en dirección X e Y.
- dim_x, dim_y : medida en mm del fotograma en dirección X e Y.

Para estimar la medida de un píxel se ingresan las dimensiones de ancho y alto de un fotograma con la ayuda de una regla, y el software realiza el resto de los cálculos.

Sin embargo si la cámara no está perfectamente perpendicular a la mesa de trabajo, la dimensión de los píxeles cambia según su posición.

Otro efecto no deseado ocurre según el tipo de lente de la cámara y la distancia al objetivo.

En la figura 3.14a se puede ver la imagen de una regla capturada con una cámara perpendicular y en la 3.14b otra capturada a cierto ángulo. Se puede apreciar que en el último caso se distorsionan las medidas notablemente.

Estas alteraciones afectan la dimensión de cada píxel e impactan directamente en el error de alineación.

Para mitigar este problema se desarrollaron dos técnicas:

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

En la figura 3.15 se aplica esta corrección a las imágenes de la figura 3.14 y se puede apreciar que en la zona reducida ahora la distorsión es menos apreciable.



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



(b) Toma en ángulo. Se puede ver la distorsión de las medidas de la regla en función de la posición en la imagen.



(c) Fotografía tomada por una lente de gran angular que resalta el efecto de distorsión.

FIGURA 3.14. Revelación del efecto de distorsión de las medidas en función del ángulo de captura de la imagen, distancia al objetivo y tipo de lente.

donde:

- $pixel_x, pixel_y$: dimensión en mm de cada píxel en dirección X e Y.
- $pixels_x, pixels_y$: número de píxeles de cada fotograma en dirección X e Y.
- dim_x, dim_y : medida en mm del fotograma en dirección X e Y.

Para estimar la medida de un píxel se ingresan las dimensiones de ancho y alto de un fotograma con la ayuda de una regla, y el software realiza el resto de los cálculos.

Sin embargo, si la cámara no está perfectamente perpendicular a la mesa de trabajo, la dimensión de los píxeles cambia según su posición.

Otro efecto no deseado ocurre según el tipo de lente de la cámara y la distancia al objetivo.

En la figura 3.14a se puede ver la imagen de una regla capturada con una cámara perpendicular y en la 3.14b otra capturada a cierto ángulo. Se puede apreciar que en el último caso se distorsionan las medidas notablemente.

Estas alteraciones afectan la dimensión de cada píxel e impactan directamente en el error de alineación.

Para mitigar este problema se desarrollaron dos técnicas:

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

En la figura 3.15 se aplica esta corrección a las imágenes de la figura 3.14 y se puede apreciar que en la zona reducida ahora la distorsión es menos apreciable.



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

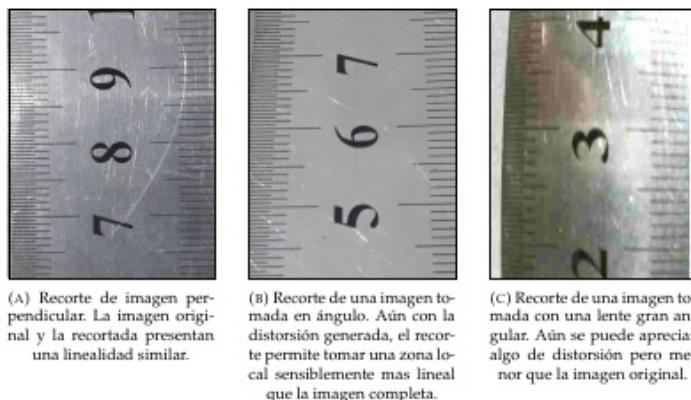


(b) Toma en ángulo. Se puede ver la distorsión de las medidas de la regla en función de la posición en la imagen.



(c) Fotografía tomada por una lente de gran angular que resalta el efecto de distorsión.

FIGURA 3.14. Revelación del efecto de distorsión de las medidas en función del ángulo de captura de la imagen, distancia al objetivo y tipo de lente.



- Centrado: se utiliza la dimensión del píxel como una aproximación, pero luego se itera el movimiento de la máquina hasta que la marca se posiciona justo en el centro de la imagen.

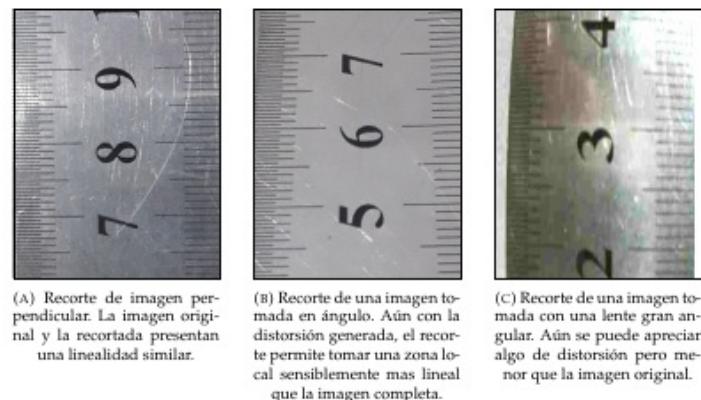
Este procedimiento aletarga el proceso de alineación, pero minimiza los efectos de error mencionados, debido a que la distancia entre el centro de la imagen y la marca tienden a cero píxeles.

En la figura 3.16 se muestra una marca a cierta distancia del centro de la imagen. Luego de varias iteraciones, se posiciona la marca justo en el centro donde los errores son despreciables.

3.7. Secuencia de pasos para alinear

En el software nkHACK se implementaron dos métodos de alineación: manual y automático.

En la tabla 3.2 se detalla el método manual con una imagen representativa de las acciones ejecutadas en cada paso.



- Centrado: se utiliza la dimensión del píxel como una aproximación, pero luego se itera el movimiento de la máquina hasta que la marca se posiciona justo en el centro de la imagen.

Este procedimiento aletarga el proceso de alineación, pero minimiza los efectos de error mencionados, debido a que la distancia entre el centro de la imagen y la marca tiende a cero píxeles.

En la figura 3.16 se muestra una marca a cierta distancia del centro de la imagen. Luego de varias iteraciones, se posiciona la marca justo en el centro donde los errores son despreciables.

3.7. Secuencia de pasos para alinear

En el software nkHACK se implementaron dos métodos de alineación: manual y automático.

En la tabla 3.2 se detalla el método manual con una imagen representativa de las acciones ejecutadas en cada paso.

3.7. Secuencia de pasos para alinear

43

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

Procedimiento	Imagen
Transformación: con las estimaciones de escalas en X e Y y el ángulo de rotación de la pieza se convierte el archivo GCode original y se eliminan las marcas.	
Se envía el nuevo archivo al controlador.	
Toma de archivos: se carga el archivo desde el USB virtual al controlador.	

3.7. Secuencia de pasos para alinear

43

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

Procedimiento	Imagen
Transformación: con las estimaciones de escalas en X e Y y el ángulo de rotación de la pieza, se convierte el archivo GCode original y se eliminan las marcas.	
Se envía el nuevo archivo al controlador.	
Toma de archivos: se carga el archivo desde el USB virtual al controlador.	

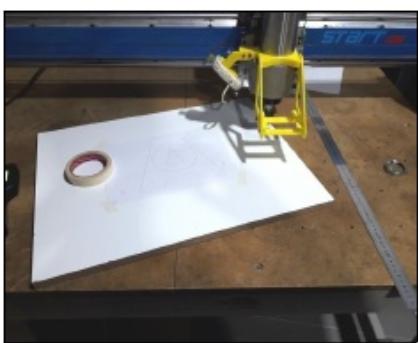


FIGURA 4.2. Placa de madera con la impresión del trabajo de corte pegada. Esto permite ubicar la pieza en diferentes posiciones y probar los resultados del sistema de alineación.

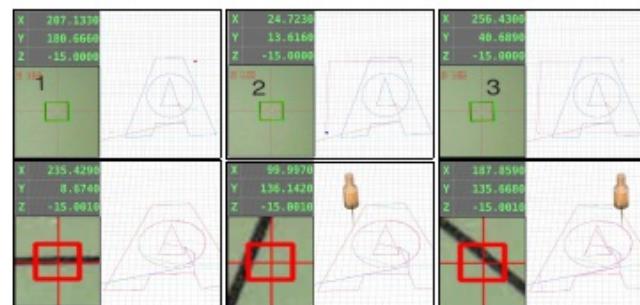


FIGURA 4.3. Secuencia de pasos para los ensayos de corte simulado. Se utiliza el recuadro rojo en el centro de la imagen como testigo del error máximo.

TABLA 4.1. Información recolectada durante repetidos ensayos a un mismo diseño de corte pero posicionado en diferentes ángulos y desplazamientos.

Ángulo	Delta X	Delta Y	Escala X	Escala Y	Error máximo
15,46	0	0	1	1	0,4
-17,7	-2,54	-5,46	1	1	0,6
28,07	-3,76	6,83	1	1	0,65
-25,2	-0,54	-6,79	1	1	0,4
1,71	-0,01	-0,01	1	1	0,35

Sin embargo durante el análisis se determinó que tanto la mesa de corte utilizada como la impresora láser tienen distorsiones no lineales.

Por ejemplo, la mesa de corte no respeta las medidas a lo largo de todo el eje X. Hay ciertas zonas con mayor error que otras.

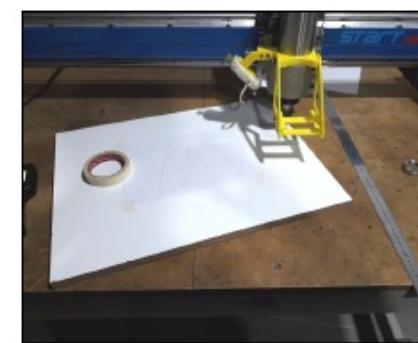


FIGURA 4.2. Placa de madera con la impresión del trabajo de corte pegada. Esto permite ubicar la pieza en diferentes posiciones y probar los resultados del sistema de alineación.

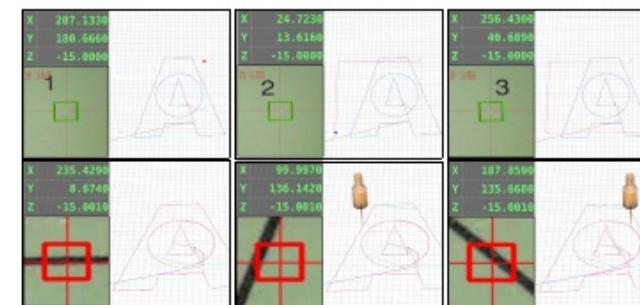


FIGURA 4.3. Secuencia de pasos para los ensayos de corte simulado. Se utiliza el recuadro rojo en el centro de la imagen como testigo del error máximo.

TABLA 4.1. Información recolectada durante repetidos ensayos a un mismo diseño de corte pero posicionado en diferentes ángulos y desplazamientos.

Ángulo [grados]	Delta X [mm]	Delta Y [mm]	Escala X [mm]	Escala Y [mm]	Error máximo [mm]
15,46	0	0	1	1	0,4
-17,7	-2,54	-5,46	1	1	0,6
28,07	-3,76	6,83	1	1	0,65
-25,2	-0,54	-6,79	1	1	0,4
1,71	-0,01	-0,01	1	1	0,35

Sin embargo durante el análisis se determinó que tanto la mesa de corte utilizada como la impresora láser tienen distorsiones no lineales.

También se encontró que el sistema es muy susceptible a variaciones de altura del objeto a cortar. Es más notable cuando se trata de una simulación, dado que en la cámara estos efectos se ven amplificados.

Se presume que para lograr reducir los errores de corte se requiere de elementos de mayor precisión para las mediciones, un ajuste fino tanto a la mesa de corte y la calibración de la impresora láser.

Con dichas herramientas se espera poder desacoplar el error de la trigonometría de alineación, si lo tuviera, de los problemas mecánicos.

4.2.3. Ensayos con impresiones escaladas

Para validar la característica de escalado del software se generó un trabajo de corte que consiste en una figura de $150\text{mm} \times 150\text{mm}$ como se muestra en la figura 4.4a y una versión a escala cuyas medidas son $140\text{mm} \times 130\text{mm}$, como se muestra en la figura 4.4b.

En el primer ensayo se procede a cortar el archivo original con su archivo de corte GCode correcto.

En el segundo ensayo se mantiene el archivo GCode, pero se intenta cortar el perímetro impreso en escala reducida.

En la figura 4.5 se puede ver parte del proceso de reconocimiento y simulación para el trazo original y en la figura 4.6 para el trazo escalado.

En este caso se aprecia que las marcas están mucho más distantes que lo que deberían debido a la escala, pero aun así, son reconocidas y el software lo corrige.

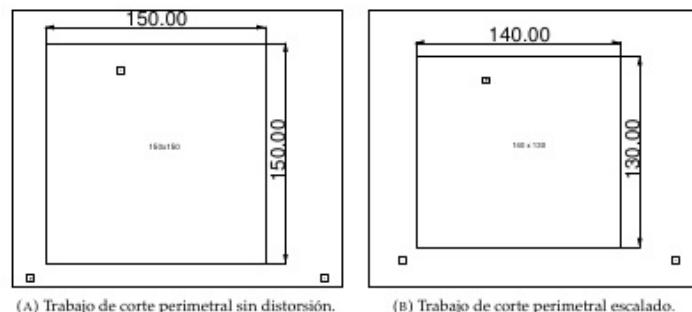


FIGURA 4.4. Generación de un trabajo de corte perimetral y otro escalado con el objetivo de validar la función de escalado no lineal del software.

En la tabla 4.2 se comparan los resultados del proceso de corte para el cuadrado sin distorsión y su contraparte escalada.

Se puede ver que los resultados son prácticamente equivalentes aún cuando las escalas reflejan la diferencia.

Por ejemplo, la mesa de corte no respeta las medidas a lo largo de todo el eje X. Hay ciertas zonas con mayor error que otras.

También se encontró que el sistema es muy susceptible a variaciones de altura del objeto a cortar. Es más notable cuando se trata de una simulación, dado que en la cámara estos efectos se ven amplificados.

Se presume que para lograr reducir los errores de corte se requiere de elementos de mayor precisión para las mediciones, un ajuste fino tanto a la mesa de corte y la calibración de la impresora láser.

Con dichas herramientas se espera poder desacoplar el error de la trigonometría de alineación, si lo tuviera, de los problemas mecánicos.

4.2.3. Ensayos con impresiones escaladas

Para validar la característica de escalado del software se generó un trabajo de corte que consiste en una figura de $150\text{mm} \times 150\text{mm}$ como se muestra en la figura 4.4a y una versión a escala cuyas medidas son $140\text{mm} \times 130\text{mm}$, como se muestra en la figura 4.4b.

En el primer ensayo se procede a cortar el archivo original con su archivo de corte GCode correcto.

En el segundo ensayo se mantiene el archivo GCode, pero se intenta cortar el perímetro impreso en escala reducida.

En la figura 4.5 se puede ver parte del proceso de reconocimiento y simulación para el trazo original y en la figura 4.6 para el trazo escalado.

En este caso se aprecia que las marcas están mucho más distantes que lo que deberían debido a la escala, pero aun así, son reconocidas y el software lo corrige.

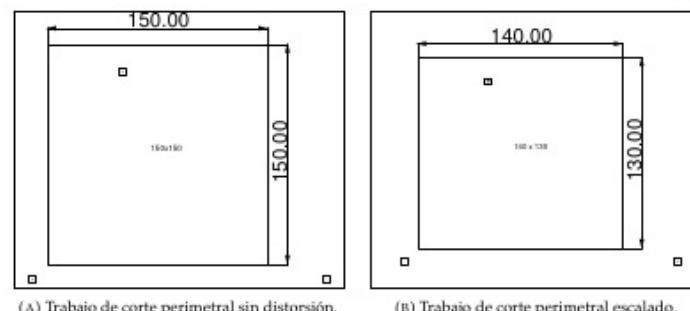


FIGURA 4.4. Generación de un trabajo de corte perimetral y otro escalado con el objetivo de validar la función de escalado no lineal del software.

En la tabla 4.2 se comparan los resultados del proceso de corte para el cuadrado sin distorsión y su contraparte escalada.

Se puede ver que los resultados son prácticamente equivalentes aún cuando las escalas reflejan la diferencia.

4.2. Pruebas funcionales del hardware

49

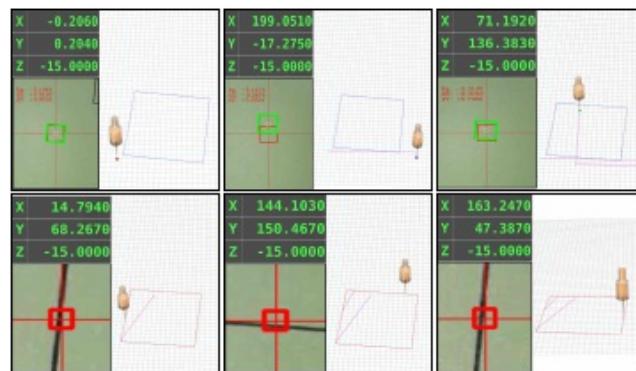


FIGURA 4.5. Secuencia de pasos para la simulación de corte perimetral con su respectivo archivo de corte GCode sin distorsión.

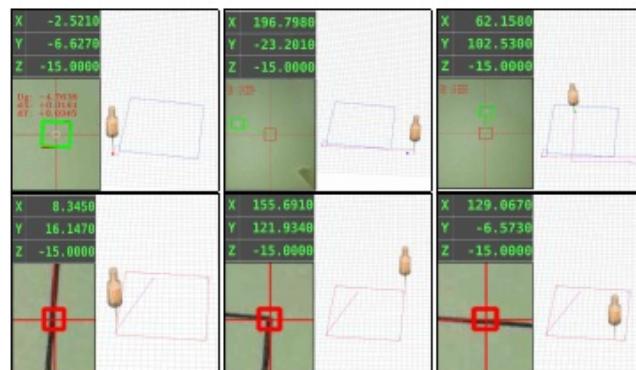


FIGURA 4.6. Secuencia de pasos para la simulación de corte perimetral escalado con el archivo de corte GCode de la versión sin escalar. Se puede notar que las marcas dos y tres aparecen distantes del lugar esperado debido al escalado.

TABLA 4.2. Información recolectada durante dos ensayos de corte de un perímetro sin distorsión y otro escalado.

Ángulo	Delta X	Delta Y	Escala X	Escala Y	Error máximo
-4,00	-0,18	0,2	1	1	0,3
-4,00	-2,51	-6,59	0,93	0,87	0,3

4.2. Pruebas funcionales del hardware

49

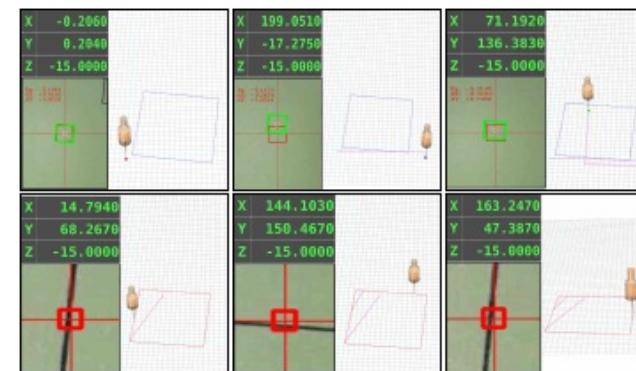


FIGURA 4.5. Secuencia de pasos para la simulación de corte perimetral con su respectivo archivo de corte GCode sin distorsión.

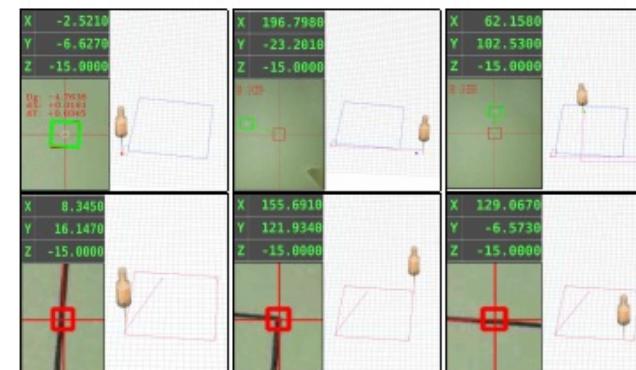


FIGURA 4.6. Secuencia de pasos para la simulación de corte perimetral escalado con el archivo de corte GCode de la versión sin escalar. Se puede notar que las marcas dos y tres aparecen distantes del lugar esperado debido al escalado.

TABLA 4.2. Información recolectada durante dos ensayos de corte de un perímetro sin distorsión y otro escalado.

Ángulo [grados]	Delta X [mm]	Delta Y [mm]	Escala X [mm]	Escala Y [mm]	Error máximo [mm]
-4,00	-0,18	0,2	1	1	0,3
-4,00	-2,51	-6,59	0,93	0,87	0,3

4.2.4. Ensayos con diferentes resoluciones de vídeo

Si bien cuanto más resolución tenga la cámara más pequeño será el tamaño del pixel, esta mejora parece no trasladarse linealmente a los resultados para las dimensiones de marcas utilizadas.

Por el contrario, una mayor resolución en la imagen implica mayor tiempo de procesamiento.

Como la PocketBeagle no es especialmente eficiente para procesar imágenes, se buscó lograr un punto óptimo para la resolución de las imágenes.

En la tabla 4.3 se muestra una comparativa de 3 resoluciones típicas de video y para cada caso se probaron dos niveles de compresión.

TABLA 4.3. Comparación de diferentes resoluciones de imágenes y sus resultados para la identificación de marcas.

Ancho x alto	Calidad [%]	Delta X [mm]	Delta Y [mm]	Ángulo [grados]	Imagen
240x320	1 %	5,04	-5,09	18,30	
240x320	50 %	4,98	-5,04	19,48	
640x480	1 %	5,04	-5,05	18,90	
640x480	50 %	5,02	-5,07	19,48	

4.2.4. Ensayos con diferentes resoluciones de vídeo

Si bien cuanto más resolución tenga la cámara más pequeño será el tamaño del pixel, esta mejora parece no trasladarse linealmente a los resultados para las dimensiones de marcas utilizadas.

Por el contrario, una mayor resolución en la imagen implica mayor tiempo de procesamiento.

Como la PocketBeagle no es especialmente eficiente para procesar imágenes, se buscó lograr un punto óptimo para la resolución de las imágenes.

En la tabla 4.3 se muestra una comparativa de 3 resoluciones típicas de video y para cada caso se probaron dos niveles de compresión.

TABLA 4.3. Comparación de diferentes resoluciones de imágenes y sus resultados para la identificación de marcas.

Ancho x alto [pixeles]	Calidad [%]	Delta X [mm]	Delta Y [mm]	Ángulo [grados]	Imagen
240x320	1	5,04	-5,09	18,30	
240x320	50	4,98	-5,04	19,48	

4.2. Pruebas funcionales del hardware

51

TABLA 4.3. Comparación de diferentes resoluciones de imágenes y sus resultados para la identificación de marcas. Continuación.

Ancho x alto	Calidad	Delta X	Delta Y	Ángulo	Imagen
960x720	1 %	5,05	-5,06	18,90	
960x720	50 %	5,01	-5,07	19,48	

Se puede concluir que no es tan relevante la resolución de la imagen como la calidad de compresión. Es preferible una calidad media con una resolución baja a una resolución alta con una compresión baja.

Para la mayoría de los ensayos de este trabajo se utilizó una resolución de 640x480 con una calidad del 20 % al 30 %. Estos valores permiten una buena calidad de inspección sin ralentizar el procesamiento.

4.2. Pruebas funcionales del hardware

51

TABLA 4.3. Comparación de diferentes resoluciones de imágenes y sus resultados para la identificación de marcas. Continuación.

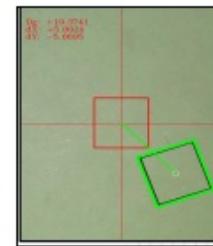
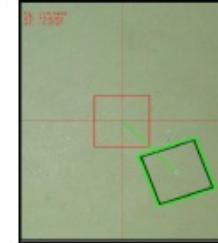
Ancho x alto [pixe-les]	Calidad [%]	Delta X [mm]	Delta Y [mm]	Ángulo [gra-dos]	Imagen
640x480	1	5,04	-5,05	18,90	
640x480	50	5,02	-5,07	19,48	
960x720	1	5,05	-5,06	18,90	

TABLA 4.3. Comparación de diferentes resoluciones de imágenes y sus resultados para la identificación de marcas. Continuación.

Ancho x alto [pixe- les]	Calidad [%]	Delta X [mm]	Delta Y [mm]	Ángulo [gra- dos]	Imagen
960x720	50	5,01	-5,07	19,48	

Se puede concluir que no es tan relevante la resolución de la imagen como la calidad de compresión. Es preferible una calidad media con una resolución baja a una resolución alta con una compresión baja.

Para la mayoría de los ensayos de este trabajo se utilizó una resolución de 640x480 con una calidad del 20 % al 30 %. Estos valores permiten una buena calidad de inspección sin ralentizar el procesamiento.

Capítulo 5

Conclusiones

En el presente capítulo se presentan las conclusiones generales del trabajo, algunas consideraciones sobre las herramientas utilizadas y los próximos pasos.

5.1. Conclusiones generales

En la siguiente lista se detallan los objetivos cumplidos más destacados y se comentan algunas dificultades, desafíos y técnicas aplicadas en cada punto:

- Se logró el objetivo de reconocer marcas no solo con una cámara de video sino también con el uso de un celular. Esto amplía el mercado de usuarios posibles y facilita la integración en máquinas existentes con mínimo costo y esfuerzo.
- Se implementó un software de control que no solo permite alinear el trabajo a cortar sino que también simula la máquina en tiempo real. Esto es de gran valor dado que permite visualizar posibles problemas en el corte y analizar la disposición del material en la mesa de trabajo con antelación.
- Se utilizó una de las plataformas más pequeñas y económicas del mercado, y aun así queda margen para actualizaciones y mejoras.
- Si bien el equipo elegido no cuenta con una interfaz para manejarlo remoto, la técnica utilizada, permite mantener el equipo original intacto, y compartir el mando a distancia junto con el acceso web. Esta característica permite probar el equipo en una máquina funcionando en solo unos minutos y sin intervención.
- Se logró el objetivo de poder acceder a la máquina remotamente, tanto desde la web como desde una conexión segura SSH.
- Se logró realizar el desarrollo del software con tecnología web y facilitar su uso en desde cualquier sistema operativo y dispositivo, incluso desde una tableta o móvil.
- Se lograron precisiones de corrección en promedio por debajo de 0,5mm. Si bien es un punto a mejorar, se considera aceptable considerando las calidades de las herramientas utilizadas.
- Se encontró una dificultad importante en el desarrollo del driver SPI que tomó considerablemente más tiempo **que el** esperado.

Capítulo 5

Conclusiones

En el presente capítulo se presentan las conclusiones generales del trabajo, algunas consideraciones sobre las herramientas utilizadas y los próximos pasos.

5.1. Conclusiones generales

En la siguiente lista se detallan los objetivos cumplidos más destacados y se comentan algunas dificultades, desafíos y técnicas aplicadas en cada punto:

- Se logró el objetivo de reconocer marcas no solo con una cámara de video sino también con el uso de un celular. Esto amplía el mercado de usuarios posibles y facilita la integración en máquinas existentes con mínimo costo y esfuerzo.
- Se implementó un software de control que no solo permite alinear el trabajo a cortar sino que también simula la máquina en tiempo real. Esto es de gran valor dado que permite visualizar posibles problemas en el corte y analizar la disposición del material en la mesa de trabajo con antelación.
- Se utilizó una de las plataformas más pequeñas y económicas del mercado, y aun así queda margen para actualizaciones y mejoras.
- Si bien el equipo elegido no cuenta con una interfaz para manejarlo remoto, la técnica utilizada, permite mantener el equipo original intacto, y compartir el mando a distancia junto con el acceso web. Esta característica permite probar el equipo en una máquina funcionando en solo unos minutos y sin intervención.
- Se logró el objetivo de poder acceder a la máquina remotamente, tanto desde la web como desde una conexión segura SSH.
- Se logró realizar el desarrollo del software con tecnología web y facilitar su uso en desde cualquier sistema operativo y dispositivo, incluso desde una tableta o móvil.
- Se lograron precisiones de corrección en promedio por debajo de 0,5mm. Si bien es un punto a mejorar, se considera aceptable considerando las calidades de las herramientas utilizadas.
- Se encontró una dificultad importante en el desarrollo del driver SPI que tomó considerablemente más tiempo **del** esperado.

Pero dado que todo el trabajo se basa en la eficiencia de dicho software fue necesario realizar uno a medida y afrontar el retraso.

A excepción de este driver, el resto de las tareas se alcanzaron en los plazos esperados.

- Se aprovechó la tecnología de configFS de Linux para cumplir con el requisito de realizar la transferencia de archivos remoto.

Este punto por si solo resuelve un problema recurrente de intercambiar archivos desde una PC con el uso de una unidad de almacenamiento externa.

- Se logró montar el prototipo de pruebas en una máquina CNC real y validar todas las funcionalidades del sistema.
- Se logró un algoritmo de reconocimiento muy potente capaz de reconocer marcas con cierta distorsión pero sin consumir todos los recursos del procesador.

Para lograrlo se aplicaron técnicas como: limitar los cuadros por segundo a procesar, bajar las resoluciones de imagen y limitar la cantidad de contornos permitidos por cuadro.

- Además de reconocer las marcas y alinear la pieza, se cumplió con el objetivo de escalar el archivo de corte de manera no proporcional en X e Y. Esta es una característica muy poco frecuente en soluciones similares.
- El software permite al usuario visualizar el modelo 3D de su máquina, y mover independientemente cada eje.

Esto no era un requisito, pero se lo identificó como de mucho valor y muy factible para el tiempo disponible.

- Una de las técnicas más útiles en la segunda mitad del trabajo fue haber logrado trabajar en una PC e interactuar con la PocketBeagle remotamente para el acceso a los drivers.

Esto permitió acelerar el desarrollo notablemente.

5.2. Próximos pasos

- Agregar soporte para las versiones más avanzadas del NK105 que también ofrece el fabricante y como comparten el mismo controlador, son perfectamente compatibles con este desarrollo.
- Validar el sistema mecánico de la máquina CNC y revisar la calibración de la impresora láser.

De esta manera se podrán desacoplar los errores cometidos por la mecánica y por el sistema de alineación.

- Agregar alguna protección de acceso múltiple para que varios usuarios puedan compartir el software y monitorear la máquina sin poner en riesgo la integridad del trabajo en curso.
- Simplificar la interfaz web para facilitar el uso de las funciones más recurrentes.

Pero dado que todo el trabajo se basa en la eficiencia de dicho software fue necesario realizar uno a medida y afrontar el retraso.

A excepción de este driver, el resto de las tareas se alcanzaron en los plazos esperados.

- Se aprovechó la tecnología configFS de Linux para cumplir con el requisito de realizar la transferencia de archivos remoto.

Este punto por si solo resuelve un problema recurrente de intercambiar archivos desde una PC con el uso de una unidad de almacenamiento externa.

- Se logró montar el prototipo de pruebas en una máquina CNC real y validar todas las funcionalidades del sistema.
- Se logró un algoritmo de reconocimiento muy potente capaz de reconocer marcas con cierta distorsión pero sin consumir todos los recursos del procesador.

Para lograrlo se aplicaron técnicas como: limitar los cuadros por segundo a procesar, bajar las resoluciones de imagen y limitar la cantidad de contornos permitidos por cuadro.

- Además de reconocer las marcas y alinear la pieza, se cumplió con el objetivo de escalar el archivo de corte de manera no proporcional en X e Y. Esta es una característica muy poco frecuente en soluciones similares.

- El software permite al usuario visualizar el modelo 3D de su máquina, y mover independientemente cada eje.

Esto no era un requisito, pero se lo identificó como de mucho valor y muy factible para el tiempo disponible.

- Una de las técnicas más útiles en la segunda mitad del trabajo fue haber logrado trabajar en una PC e interactuar con la PocketBeagle remotamente para el acceso a los drivers.

Esto permitió acelerar el desarrollo notablemente.

5.2. Próximos pasos

- Agregar soporte para las versiones más avanzadas del NK105 que también ofrece el fabricante y como comparten el mismo controlador, son perfectamente compatibles con este desarrollo.
- Validar el sistema mecánico de la máquina CNC y revisar la calibración de la impresora láser.

De esta manera se podrán desacoplar los errores cometidos por la mecánica y por el sistema de alineación.

- Agregar alguna protección de acceso múltiple para que varios usuarios puedan compartir el software y monitorear la máquina sin poner en riesgo la integridad del trabajo en curso.
- Simplificar la interfaz web para facilitar el uso de las funciones más recurrentes.

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] <https://www.pv-automation.com/en/>.
- [6] <https://www.pv-automation.com>.
- [7] [https://www.machinokit.io](https://www.machinakit.io).
- [8] <https://www.summa.com/en/solutions/>.
- [9] <https://www.weihong.com.cn/en/products/controller/20160424/23.html>.
- [10] *Ecosistema de placas de desarrollo de proposito general*. <https://http://beagleboard.org/>. (Visitado 02-02-2021).
- [11] <https://www.xilinx.com/products/boards-and-kits/1-hydd4z.html>.
- [12] <https://www.python.org/>.
- [13] <https://docs.python.org/3/library/asyncio.html>.
- [14] <https://docs.aiohttp.org/en/stable/>.
- [15] <https://en.wikipedia.org/wiki/HTML5>.
- [16] <https://en.wikipedia.org/wiki/CSS>.
- [17] <https://en.wikipedia.org/wiki/JavaScript>.
- [18] <https://get.webgl.org/>.
- [19] <https://https://threejs.org/>.
- [20] <https://en.wikipedia.org/wiki/WebSocket>.
- [21] <https://socket.io/>.
- [22] <https://opencv.org/>.
- [23] <https://pypi.org/project/pyopencv/>.
- [24] <https://play.google.com/store/apps/details?id=com.pas.websocket&hl=en&gl=US>.
- [25] Christopher Hallinan. *Embedded Linux Primer, Second Edition*. A Practical, Real-World Approach. 2010.
- [26] Alessandro Rubini by Jonathan Corbet y Greg Kroah-Hartman. *Linux Device Drivers, Third Edition*. 6556. Third Edition. 2005.
- [27] https://www.kernel.org/doc/Documentation/usb/gadget_configfs.txt.

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/](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.machinokit.io](https://www.machinikit.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/>.
- [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.websocket&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_configfs.txt.