DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA SCADA DISTRIBUIDO PARA EL MONITOREO Y CONTROL DE CELDAS DE MANUFACTURA CNC

Universidad Nacional de Colombia

FACULTAD DE INGENIERÍA y ARQUITECTURA
Sede Manizales

PRESENTADO POR:

Juan Manuel Mejia Vasco
Lisesth Carolina Yela Medicis
Maria Isabel Aristizabal Medina
Juan David Lopez Rios

PRESENTADO A:
PROF. JUAN BERNARDO GÓMEZ-MENDOZA

ASIGNATURA:
EMBEDDED LINUX SYSTEM PROGRAMMING (2025-2S)

Manizales, Caldas, COLOMBIA
DICIEMBRE, 2025

Índice

1. Planteamiento del Problema, Visión General y Requisitos

1.1. Definición del Problema y Motivación

En el desarrollo de dispositivos electrónicos, el prototipado rápido se vuelve una necesidad en el momento en que se quiere proponer un producto final que solucione un problema. Aunque existen servicios externos de fabricación que ofrecen alta precisión y acabado profesional, estos suelen implicar tiempos de entrega prolongados y costos elevados, lo que dificulta realizar correcciones de forma rápida, principalmente en la parte funcional de los prototipos. Por esto, una alternativa a estos métodos de fabricación es el uso de ruteadoras CNC para el fresado de PCB, ya que representa una alternativa ágil y económica para la fabricación de prototipos, permitiendo obtener resultados en una fracción del tiempo que toma con empresas de fabricación especializadas y sin depender de proveedores externos.
Sin embargo, cuando la cantidad de placas a fabricar aumenta o se necesita producir varios prototipos al mismo tiempo, una sola ruteadora CNC deja de ser suficiente para cumplir con los tiempos y números requeridos. En estos casos, utilizar varias ruteadoras trabajando en paralelo se convierte en una alternativa práctica para aumentar la capacidad de producción y acercar el proceso a una fabricación en serie. Este enfoque permite repartir el trabajo entre distintas máquinas, aprovechando mejor el tiempo disponible y reduciendo los periodos de espera.
No obstante, al incorporar más equipos, también aparecen nuevos desafíos relacionados con la coordinación y supervisión del proceso, ya que cada máquina debe ser controlada y monitoreada adecuadamente para mantener la eficiencia, la calidad y la trazabilidad de la producción. Dicho esto, en un escenario en el que se realiza producción de PCB mediante varias ruteadoras CNC, también aparecen problemas relacionados con la coordinación y supervisión del proceso, ya que cada máquina debe ser controlada y monitoreada para asegurar una salida satisfactoria del corte.

El proceso de control de cada una de las máquinas se realiza de manera individual por medio de dos métodos principales:

- **I**nserción manual de una tarjeta SD que contiene el archivo de ruteado (G-code) previamente cargado desde un computador externo.
- **C**onexión serial directa entre la máquina y un PC, donde un programa "G-code sender" envía en tiempo real las instrucciones de movimiento y mecanizado.

En cualquiera de los dos casos, el proceso requiere una intervención humana directa o la dependencia de un PC dedicado por máquina, lo cual limita la escalabilidad y eficiencia del sistema. Esta arquitectura descentralizada implica una mayor cantidad de recursos humanos y equipos de cómputo dedicados al control de las máquinas, incrementando los costos operativos, el consumo energético y los tiempos de preparación por lote de producción. Además, la intervención humana repetitiva introduce un mayor riesgo de error tanto en la carga de archivos como en la ejecución del proceso, pudiendo causar fallos de mecanizado, pérdida de material o inconsistencias entre placas. Por otra parte, el seguimiento del proceso productivo presenta limitaciones significativas: el tiempo de ejecución de cada trabajo, los estados de

las máquinas (en operación, en espera, en error) y la trazabilidad de cada PCB fabricada no se registran automáticamente, sino de forma manual o parcial, lo cual dificulta la obtención de métricas de desempeño, la planificación de la producción y la detección temprana de fallas. En consecuencia, la ausencia de un sistema centralizado de control y monitoreo impide coordinar eficientemente el conjunto de ruteadoras CNC, y obstaculiza la integración con plataformas de análisis de datos o de gestión de producción. Ante esta problemática, surge la necesidad de diseñar e implementar una arquitectura de control distribuido con gestión centralizada, que permita la coordinación, supervisión y registro automatizado del estado de múltiples máquinas CNC, reduciendo la intervención humana, el número de equipos intermedios, y mejorando la eficiencia operativa y trazabilidad del proceso.

La Solución Propuesta: Diseñar e implementar una arquitectura de control distribuido (local y remoto) con gestión centralizada, que permita la coordinación, supervisión y registro automatizado del estado de múltiples máquinas CNC, reduciendo la intervención humana, el número de equipos intermedios, y mejorando la eficiencia:

- **Nodos de Control (Edge):** Se encargan estrictamente de la ejecución en tiempo real y seguridad de la máquina.
- **HMI Centralizado (Gateway):** Permite la supervisión remota y el despacho de tareas a múltiples máquinas desde un único punto de acceso.

1.2. Arquitectura del Sistema y Justificación (Trade-offs)

El sistema utiliza una arquitectura distribuida Maestro-Esclavo sobre una red local WiFi, extendida hacia la nube para telemetría segura.

1.2.1. Hardware y Decisiones de Diseño

- **HMI Gateway (Maestro) - Raspberry Pi 3 Model B:** Inicialmente se planteó el uso de una Lichee RV (RISC-V). Sin embargo, durante la fase de desarrollo, se detectó una inmadurez crítica en los drivers de video y documentación limitada para interfaces gráficas. Se tomó la decisión estratégica de migrar a **Raspberry Pi 3** para garantizar una plataforma Linux estable (Debian), soporte nativo HDMI y una pila de red robusta, permitiendo enfocar el esfuerzo en la lógica de la aplicación distribuida.
- **Implementación de Control CNC vía WebSocket: Integración entre Raspberry Pi y FluidNC** En el ecosistema de FluidNC, el protocolo WebSocket actúa como la columna vertebral de la comunicación, estableciendo un túnel bidireccional y persistente sobre la red WiFi que reemplaza a la conexión serial tradicional. Esta tecnología permite la transmisión fluida de comandos G-code y la recepción simultánea de reportes de estado (posición, alarmas) con una latencia mínima, garantizando que el controlador ESP32 pueda recibir instrucciones y enviar feedback en tiempo real sin la sobrecarga de abrir y cerrar conexiones repetidamente, algo crítico para la precisión del control numérico.

Aprovechando esta arquitectura, en nuestro proyecto implementamos una función específica en lenguaje C dentro de una Raspberry Pi, diseñada para actuar como un cliente WebSocket automatizado. Esta implementación establece un puente directo con la ESP32, encargándose de encapsular las instrucciones de movimiento y decodificar las respuestas del hardware; de esta forma, logramos una comunicación robusta y de alto rendimiento que integra la lógica de control de la Raspberry con la ejecución física de la máquina, prescindiendo de interfaces web humanas para la operación.

- **Nodo CNC (Esclavo) - ESP32:** Se seleccionó por su arquitectura Dual Core, permitiendo dedicar un núcleo a la comunicación WiFi/HTTP y el otro a la generación de pulsos de motor en tiempo real estricto.

1.2.2. Nivel de Nube y Persistencia (Cloud Layer)

Para garantizar la escalabilidad, la seguridad y el acceso remoto, se integraron servicios web avanzados:

- **AWS EC2:** Aloja el Backend administrativo para visualización remota.
- **Comunicación Segura (HTTPS/TLS):** Toda la comunicación entre el Gateway (Raspberry Pi) y la Nube (AWS) se realiza mediante el protocolo **HTTPS**. Esto asegura que las órdenes de producción y los datos de telemetría viajen cifrados (TLS 1.2+), protegiendo la integridad de la información contra ataques de interceptación en redes industriales.
- **Base de Datos MySQL:** Gestiona la persistencia de las órdenes de producción. A diferencia de mantener estados en memoria, esto asegura la recuperación de la "Cola de Producción" ante reinicios del sistema.
- **Lógica de Sincronización:** La Raspberry Pi realiza peticiones seguras a la API REST para actualizar dinámicamente la lista de tareas disponibles en la pantalla del operario.

1.3. Requisitos Cuantificables del Sistema

ID	Requisito	Criterio de Aceptación (Testable)
REQ-F-01	**Monitoreo Multi-Instancia:** El HMI debe gestionar múltiples nodos simultáneamente.	El sistema muestra actualizaciones de al menos **2 nodos** concurrentemente.
REQ-F-02	**Persistencia de Tareas:** Las órdenes creadas remotamente deben estar disponibles en planta.	El HMI recupera correctamente la lista de archivos G-Code desde la DB MySQL.
REQ-NF-01	**Latencia de Estado:** Visualización en tiempo casi real.	Latencia Evento (ESP32) $\rightarrow$ Pantalla (HMI) $<$ **1 segundo**.

Cuadro 1: Requisitos Funcionales y No Funcionales

2. Plan de Verificación y Pruebas Unitarias

Se diseñó un plan centrado en la integridad de datos y protocolos.

ID	Módulo / Función	Descripción	Resultado
UT-01	`Servidor AWS`	Prueba de Despliegue y Conexión SSH, Base de Datos y Almacenamiento de Archivos, Inserción de una orden de producción.	PASÓ
UT-02	`parser.c`	Filtrado de comandos G-Code seguros (G0, G1 vs M-codes prohibidos).	PASÓ
UT-03	`network.c`	Re-conexión automática tras caída de WiFi.	PASÓ

Cuadro 2: Matriz de Pruebas Unitarias (HMI Backend)

3. Pruebas de Funcionamiento e Integración del Sistema

Para validar la correcta implementación de la arquitectura IoT, se realizaron pruebas exhaustivas en el servidor desplegado en AWS EC2 (Ubuntu 20.04), verificando tanto la persistencia de datos como la comunicación en tiempo real.

3.1. Prueba de Despliegue y Conexión SSH

Se verificó la accesibilidad remota a la instancia EC2 mediante el protocolo SSH utilizando llaves de seguridad (.pem). Se configuró el firewall (Security Groups) de AWS para permitir tráfico HTTP (Puerto 80/3000) y SSH (Puerto 22).

Listing 1: Conexión exitosa al servidor

```
$ ssh -i "ESP32-webserver.pem" ubuntu@18.223.169.118
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux x86_64)
ubuntu@ip-172-31-XX-XX:~$
```

3.2. Prueba de Base de Datos y Almacenamiento de Archivos (BLOB)

Se validó la capacidad del motor MySQL para almacenar datos híbridos. A diferencia de un almacenamiento tradicional, se implementó una columna `LONGBLOB` para guardar archivos técnicos (PDFs e imágenes) directamente dentro de la base de datos, vinculados a la orden de producción.

- **Prueba:** Inserción de una orden de producción para 'Fresadora CNC' con un plano adjunto de 3MB.
- **Resultado:** La base de datos registró correctamente el binario, como se evidencia en la consulta SQL realizada en la terminal del servidor:

Listing 2: Verificación de datos en MySQL

```
mysql> SELECT id, producto, maquina, archivo_nombre FROM ordenes;
+----+----------+---------------+----------------+
| id | producto | maquina | archivo_nombre |
+----+----------+---------------+----------------+
| 6 | Eje A-20 | Fresadora CNC | plano_v1.pdf |
| 7 | Base B1 | Torno | diseno.png |
+----+----------+---------------+----------------+
```

3.3. Prueba de API REST y Descarga Selectiva (Lógica IoT)

Se diseñó y probó un script en Python para simular el comportamiento de una Raspberry Pi. El objetivo fue validar que el dispositivo pudiera:

1. Consultar la API del servidor (`GET /api/ordenes`).
2. Filtrar únicamente las órdenes asignadas a su identificador (ej. "Fresadora CNC").
3. Descargar y reconstruir el archivo binario localmente.

El resultado de la ejecución del script `gestor_maquina.py` confirmó que el dispositivo ignora órdenes ajenas y procesa correctamente las propias:

Listing 3: Salida del script Python en el cliente IoT

```
SOY LA MAQUINA: Fresadora CNC
Consultando al servidor...
El servidor tiene 2 ordenes recientes.
ENCONTRÉ UNA! Orden #6 - plano_v1.pdf
Descargando archivo del servidor...
EXITO! Archivo guardado: plano_v1.pdf
```

3.4. Prueba de Visualización en Tiempo Real (Frontend)

Finalmente, se validó la interfaz web del Dashboard. Se enviaron peticiones HTTP simuladas (mediante `curl`) para cambiar el estado de las máquinas, verificando que la interfaz reaccionara sin necesidad de recargar la página.

Se confirmó que al recibir el estado `"stopped"`, el indicador visual cambió a rojo y mostró el mensaje de error correspondiente, cumpliendo con los requisitos de monitoreo remoto.

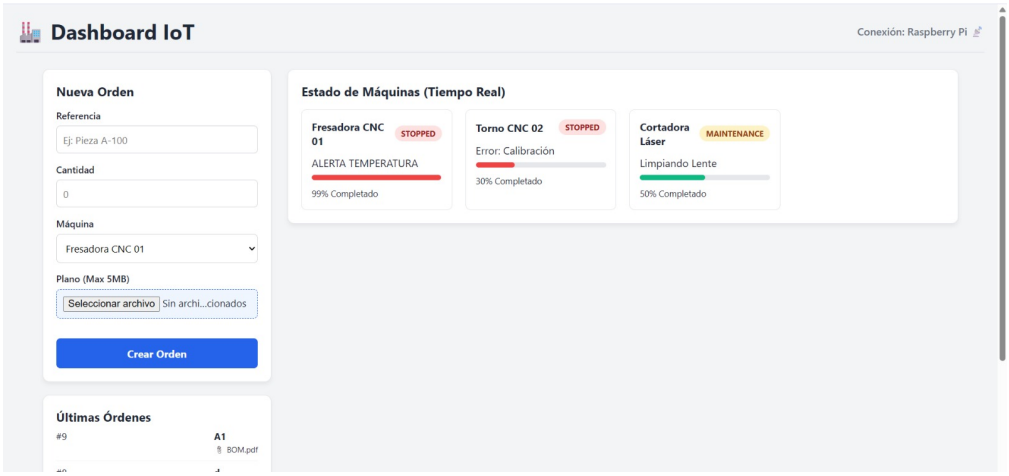

Figura 1: Interfaz Web mostrando estados en tiempo real y órdenes con adjuntos.

4. Diseño de Pruebas de Integración y Resultados

Esta sección demuestra el funcionamiento conjunto de los subsistemas: HMI + HTTP + Nodo CNC + Nube AWS.

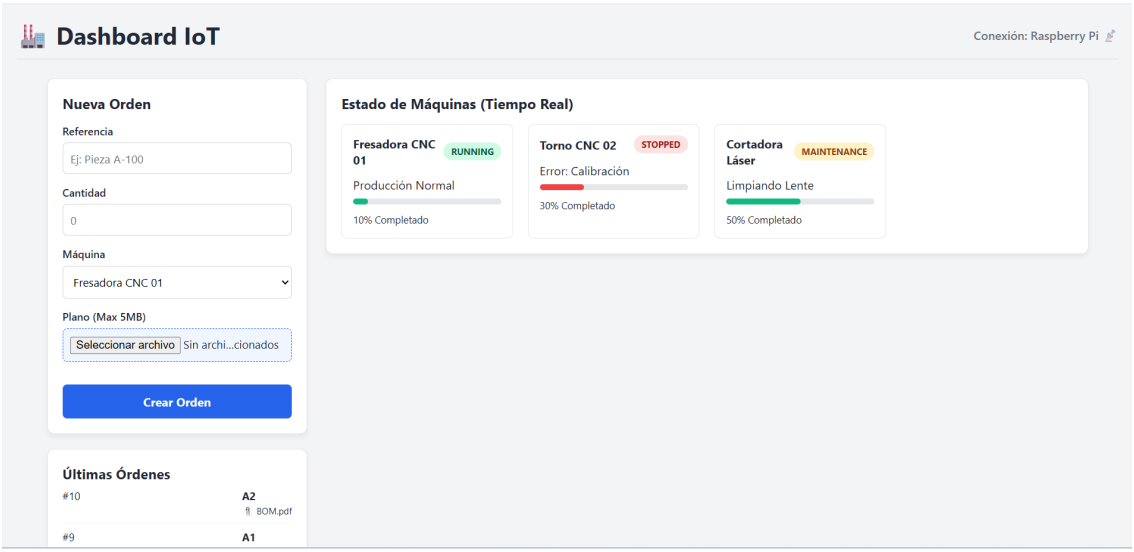

Figura 2: Interfaz deL administrador (HMI) basada en Web. El panel muestra el monitoreo en tiempo real de múltiples nodos, visualizando alertas y permitiendo la asignación dinámica de archivos G-Code.

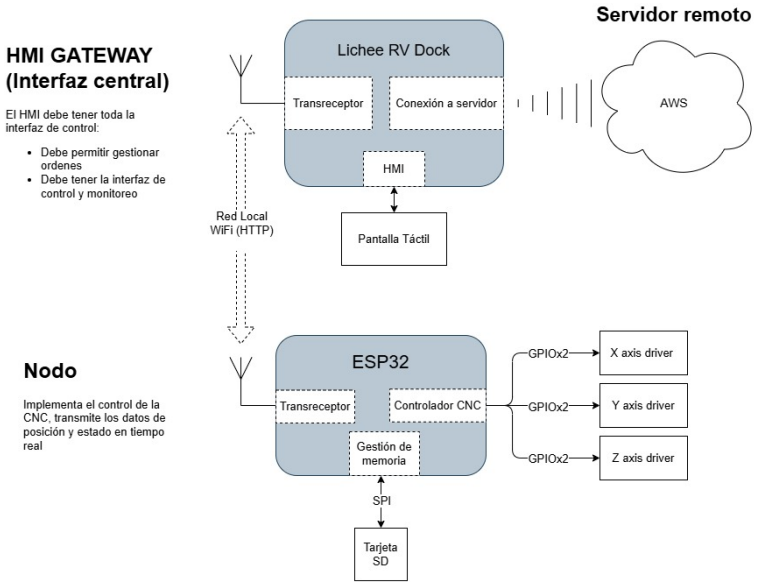

Figura 3: Arquitectura del Sistema SCADA Distribuido. Se detalla la interacción entre el Gateway Central (Raspberry Pi 3) y los Nodos de Control (ESP32) mediante MQTT y FTP.

5. Interfaz de Operación y Control

La Figura 4 presenta el Dashboard de Operación denominado "Central Control". Esta interfaz gráfica ha sido diseñada para la gestión centralizada y el control manual de múltiples máquinas CNC, segmentando la operación en tres módulos funcionales principales:

- **Panel Izquierdo (Selección de Activos):** Permite al operador conmutar dinámicamente entre las diferentes unidades conectadas (listadas como Machine 1 a 4) y gestionar la cola de trabajo mediante el botón "Agregar Tarea".
- **Panel Central (Telemetría y Estado):** Visualiza la retroalimentación en tiempo real de las coordenadas espaciales (X, Y, Z), indica el estado operativo actual (Activo/Inactivo) e incluye controles de ejecución inmediata (Play/Pause) y una consola de comandos manuales.
- **Panel Derecho (Control de Movimiento):** Proporciona una botonera digital para el desplazamiento manual (*jogging*) en los tres ejes, funciones de *homing* y un botón de parada de emergencia ("STOP MACHINE") dedicado a la unidad seleccionada, reforzado por un control de seguridad global ("STOP ALL") en la cabecera superior.

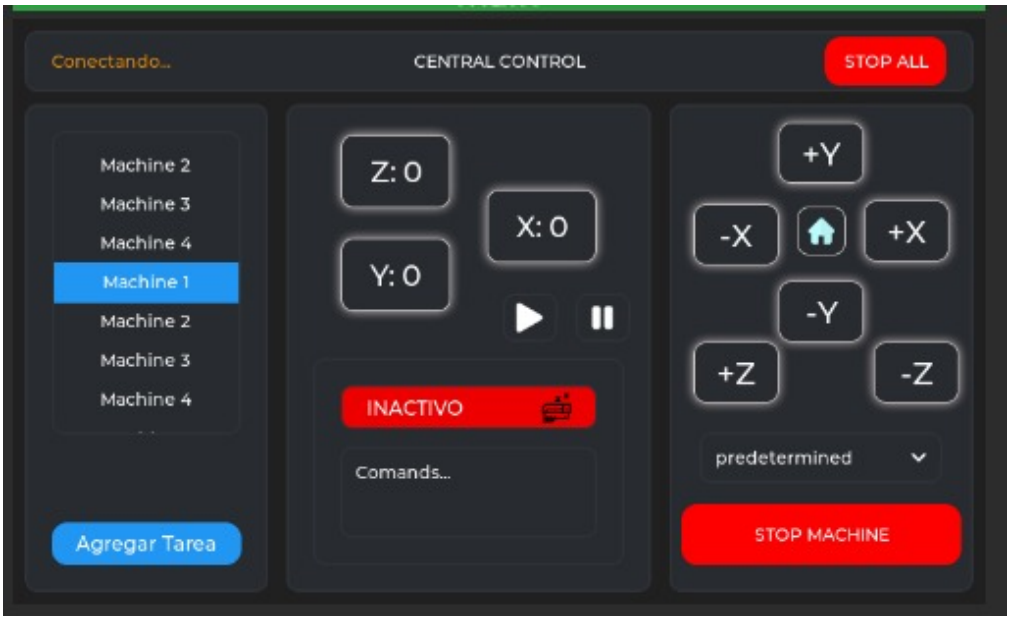

Figura 4: Interfaz "Central Control": Panel de mando para selección de máquina, monitoreo de coordenadas en tiempo real y control de movimiento.

5.1. Resultados de Pruebas End-to-End

Caso IT-01: Ciclo de Despacho Local

Se verificó la transmisión de un archivo de trabajo desde el HMI hacia el nodo físico.

Figura 5: Evidencia de Comunicación Local (Caso IT-01). La bitácora del Gateway confirma la recepción asíncrona de mensajes y la transferencia exitosa de archivos (31 KB) sin bloquear la interfaz.

Caso IT-03 y IT-04: Validación de Nube y API

Se realizaron pruebas de inyección de estados mediante `curl` y verificación de persistencia en MySQL.

Figura 6: Validación de Interoperabilidad (API REST). Prueba de inyección de estados, demostrando que el sistema procesa eventos externos JSON y actualiza el Dashboard en AWS.

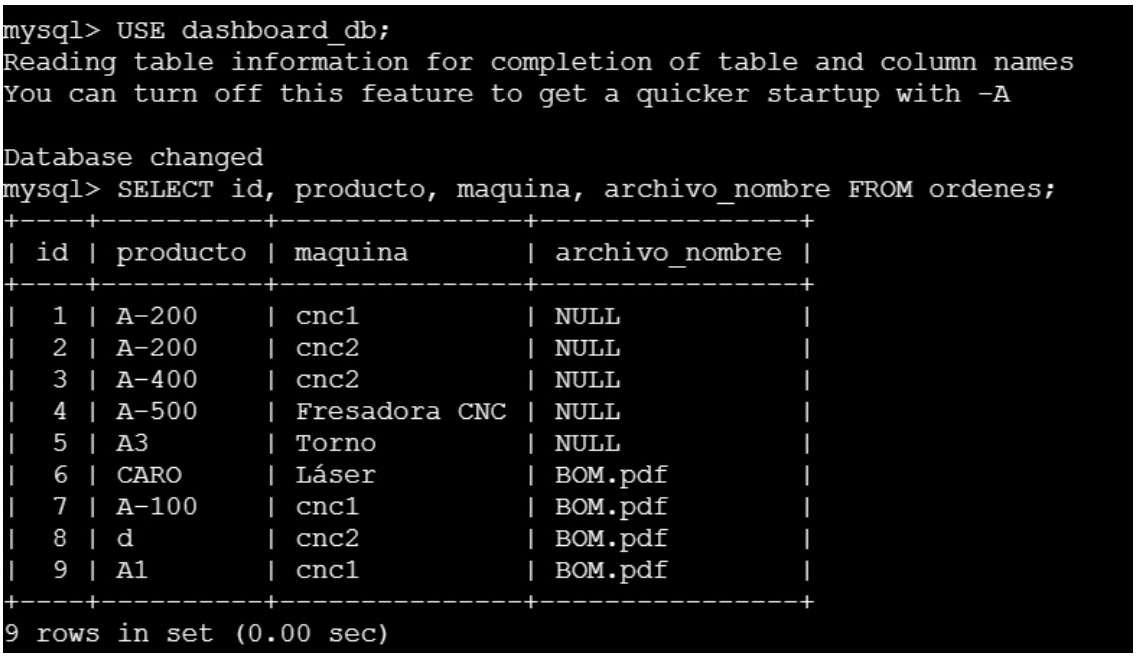

Figura 7: Persistencia de Datos. Consulta a la base de datos MySQL en AWS confirmando el registro íntegro de las órdenes de producción y su asociación con los archivos (BOM.pdf, G-Code).

6. Conclusiones y Discusión

- El desarrollo del sistema validó la arquitectura híbrida propuesta. La separación de responsabilidades entre la Raspberry Pi 3 (Gestión/Nube) y el ESP32 (Tiempo Real) resultó eficiente. La implementación de una base de datos centralizada y comunicación MQTT segura permite escalar el proyecto a un entorno de "Smart Factory" real, superando las limitaciones iniciales de hardware mediante decisiones de ingeniería justificadas.

Referencias

[1] Eclipse Foundation. *Eclipse Paho MQTT C Client Library Documentation*. Disponible en: `https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/index.html`. [Accedido: Diciembre, 2025].

[2] Raspberry Pi Foundation. *Raspberry Pi Hardware Raspbian OS Documentation*. Disponible en: `https://www.raspberrypi.com/documentation/`. [Accedido: Diciembre, 2025].

[3] Espressif Systems. *ESP-IDF Programming Guide (FreeRTOS WiFi stack)*. Disponible en: `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/`. [Accedido: Noviembre, 2025].

[4] National Institute of Standards and Technology (NIST). *The RS274NGC Interpreter - Version 3 (G-Code Standard)*. Gaithersburg, MD, 2000.

[5] Amazon Web Services. *AWS EC2 Database Migration Service Documentation*. Disponible en: `https://docs.aws.amazon.com/ec2/`. [Accedido: Diciembre, 2025].

[6] The Open Group. *POSIX.1-2017 Standard (IEEE Std 1003.1-2017)*. Sección: Threads (pthreads).

[7] B. Dring et al. *FluidNC: Next Generation CNC Firmware for ESP32*. GitHub Repository. Disponible en: `https://github.com/bdring/FluidNC`. [Accedido: Diciembre, 2025].