Práctica de Interfaz y Movimiento con Carrito Omnidireccional

Departamento de Procesos Tecnológicos e Industriales (DPTI) – ITESO

14 de octubre de 2025

Resumen

Esta práctica introduce al estudiante en la configuración de la interfaz gráfica, el control del carrito con visión computacional y la conexión con una base de datos en tiempo real. Se abordan aspectos de instalación, pestañas de configuración, y puesta en marcha del sistema de visión con marcadores ArUco.

Lista de materiales

Equipo extra

- Cámara web con cable USB
- Trípode
- Códigos ArUco
- Carrito del laboratorio
- Códigos interfaz (RoboMesha)

Estructuras mecánicas

- Perfil BOSCH de 20 mm
- Piezas impresas en 3D

Tornillería

- Tornillos M5 × 7
- Rondanas M5 × 7
- Tuercas para ranura tipo T × 7
- Tornillo M6 y tuercas M6 × 2

Cables

- Extensión USB hembra–macho de al menos 3 m

Introducción

Se explora cómo funciona una cámara a nivel físico (distancia focal, proyección de imagen, etc.), y cómo este conocimiento se aplica al control de un robot móvil con visión computacional.

Capítulo 0: Descarga de código de GitHub

El sistema requiere dos componentes principales:

- **Interfaz gráfica:** ejecutada en la computadora.
- **Código del carrito:** ejecutado en una Raspberry Pi.

Ambos se encuentran en el repositorio: `https://github.com/LuisDiazMac/Practicas-PAP/tree/main/Practicas-PAP/PRACTICA-02`

0.1 Descarga del repositorio

Para obtener el código desde GitHub, sigue los siguientes pasos:

1. Accede al enlace del repositorio.
2. En la parte superior derecha, haz clic en el botón Çode"(de color verde o gris).
3. En el menú desplegable, selecciona la opción "Download ZIP".
4. Se descargará un archivo .zip en tu computadora. Extrae el contenido en una carpeta local.

Dentro del repositorio encontrarás los scripts correspondientes tanto a la interfaz como al carrito, así como archivos de configuración como cred.json.

0.2 Configuración de entorno para la interfaz

Una vez descargado el código, es necesario instalar las dependencias de Python que requiere la interfaz gráfica para funcionar correctamente. Abre una terminal en el sistema operativo (Linux, macOS o Windows con PowerShell) y ejecuta el siguiente comando para actualizar pip e instalar las librerías necesarias:

Instalación de librerías

```
pip3 install -upgrade pip && pip3 install opencv-python numpy pillow
firebase-admin
```

Estas librerías cumplen las siguientes funciones:

- opencv-python: procesamiento de imágenes y video.
- numpy: cálculos numéricos.
- pillow: manejo de imágenes.
- firebase-admin: conexión con la base de datos en tiempo real de Firebase.

0.3 Configuración en la Raspberry Pi

El segundo componente es el script que se ejecutará en la Raspberry Pi, denominado carrito.py. Este archivo también se encuentra en el repositorio y debe ser transferido al sistema de archivos de la Raspberry, junto con el archivo cred.json.

Puedes transferir estos archivos utilizando herramientas como:

- **WinSCP (Windows)**: interfaz gráfica para transferir archivos mediante SSH.
- **scp (Linux/macOS)**: comando de terminal para copiar archivos vía SSH.
- **Cualquier otro método de tu preferencia**, siempre que respetes la estructura del proyecto.

Recomendación: crear un entorno virtual

Para mantener las dependencias del sistema organizadas y evitar conflictos con otras instalaciones, se recomienda crear un entorno virtual de Python en la Raspberry Pi. Un entorno virtual permite aislar las librerías de este proyecto del resto del sistema operativo.

Pasos para crear un entorno virtual:

1. Abre una terminal en la Raspberry Pi.
2. Asegúrate de tener instalado el módulo `venv` (suele venir por defecto con Python 3.3+).

3. Ejecuta los siguientes comandos:

```
# Crear el entorno virtual (por ejemplo, llamado venv-carrito)
python3 -m venv venv-carrito

# Activar el entorno virtual
source venv-carrito/bin/activate
```

Una vez activado el entorno virtual, notarás que el prompt de la terminal cambia, indicando que estás dentro del entorno.

Instalación de dependencias en el entorno virtual

Con el entorno activado, instala las dependencias del script del carrito ejecutando:

```
pip install numpy firebase-admin
```

Importante: Cada vez que vayas a ejecutar el script en la Raspberry, recuerda activar primero el entorno virtual con:

```
source venv-carrito/bin/activate
```

0.4 Ruta del robot

El archivo `carrito.py` contiene una variable llamada `ROBOT_ID_PATH` que indica la ruta del robot dentro de la base de datos. Esta ruta debe ser única para cada carrito y debe coincidir con la ruta configurada posteriormente en la interfaz gráfica.

Por defecto, la variable se encuentra definida así en la línea 16:

```
ROBOT_ID_PATH = "robots/123456"
```

Si deseas personalizar la ruta, puedes modificar únicamente el número (`123456`) para identificar a tu carrito. Este cambio debe mantenerse consistente en toda la práctica.

Capítulo 1: Primer acercamiento al interfaz

Una vez que se han instalado todas las librerías necesarias y se ha descargado el código, se puede proceder a ejecutar el programa sin realizar ninguna modificación en el código fuente. Para ello, simplemente se debe presionar el botón de ejecución (play) ubicado en la parte superior izquierda del entorno de desarrollo.

Descripción general de la interfaz gráfica

Al iniciar la ejecución del programa, se desplegará una ventana principal que contiene todos los elementos de la interfaz de usuario. Esta ventana puede ser dividida conceptualmente en tres secciones principales, cada una con funciones específicas:

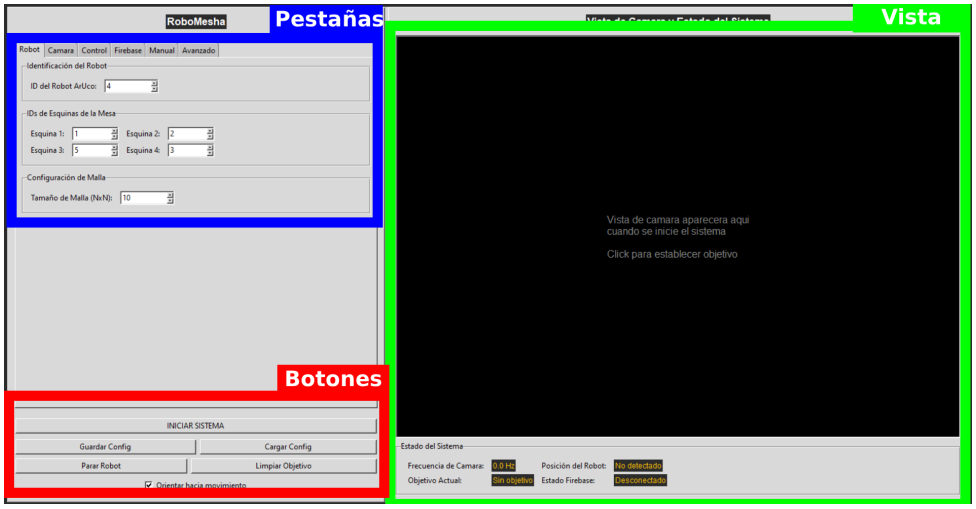

Figura 1: Secciones interfaz.

1.1. Sección de pestañas

En la parte superior de la ventana se encuentra la sección de pestañas, que permite navegar entre seis opciones diferentes. Cada pestaña está diseñada para controlar un aspecto fundamental del sistema. Las pestañas disponibles son:

- **Robot:** Configuración y monitoreo del robot móvil.
- **Cámara:** Parámetros y ajustes relacionados con la cámara de visión.
- **Control:** Herramientas para la gestión y manipulación del movimiento del robot.
- **Firebase:** Configuración y estado de la conexión con la base de datos en la nube.
- **Manual:** Opciones para control manual del robot.
- **Avanzado:** Funcionalidades adicionales y ajustes avanzados del sistema.

Cada pestaña contiene un conjunto específico de controles y visualizaciones que se abordarán detalladamente en capítulos posteriores.

1.2. Sección de vista

La sección central de la interfaz está dedicada a la visualización en tiempo real de la cámara instalada en el sistema. Aquí se transmite la imagen capturada, permitiendo observar el entorno y los códigos ArUco que sirven como referencia espacial para el robot.

Debajo de esta visualización se presentan diversos parámetros en tiempo real, tales como:

- Frecuencia de actualización de la cámara (frames por segundo).
- Posición actual del robot dentro del espacio de trabajo.
- Objetivo actual asignado al robot para su navegación.
- Estado de la conexión con la base de datos Firebase.

Estos datos permiten un monitoreo continuo y preciso del estado del sistema durante la operación.

1.3. Sección de botones principales

Finalmente, en la parte inferior o lateral de la ventana se encuentra la sección con los botones de control principales, que facilitan la interacción con el sistema. Entre las acciones disponibles se encuentran:

- Iniciar sistema: Activa el procesamiento y control del robot.
- Guardar configuración: Permite almacenar los parámetros actuales para su uso futuro.
- Cargar configuración: Carga una configuración previamente guardada.
- Detener robot: Envía una señal para detener el movimiento del robot.
- Limpiar objetivo: Resetea o elimina la meta asignada al robot.

Este primer acercamiento a la interfaz busca familiarizar al usuario con la estructura general y las funcionalidades básicas, estableciendo las bases para la realización de pruebas y configuraciones más avanzadas en capítulos posteriores.

Capítulo 2: Pestañas de Configuración

2.1Robot

La primera pestaña **Robot** permite configurar la identificación y el espacio de trabajo del sistema. Está organizada en tres secciones claramente diferenciadas:

2.1.1. Identificación del Robot

Esta sección contiene un único campo numérico ajustable:

- **ID del Robot ArUco**: Permite ingresar un número entre 0 y 50 que corresponde al marcador ArUco físicamente montado sobre el robot. Este número identifica de manera única al robot que el sistema debe rastrear y controlar. Es importante que este valor coincida exactamente con el código impreso en el marcador ArUco adherido al robot.

2.1.2. IDs de Esquinas de la Mesa

Esta sección presenta cuatro campos numéricos organizados en dos filas:

- **Esquina 1, 2, 3 y 4**: Cada campo acepta valores entre 0 y 50, correspondientes a los marcadores ArUco colocados en las cuatro esquinas del área de trabajo. Estos marcadores definen los límites del espacio donde el robot puede navegar. Los valores predeterminados son 1, 2, 5 y 3 respectivamente.

2.1.3. Configuración de Malla

Esta sección define la granularidad del espacio virtual:

- **Tamaño de Malla (N×N)**: Campo numérico que acepta valores entre 5 y 20. Este valor divide el espacio de trabajo en una cuadrícula de $N \times N$. Por ejemplo, un valor de 10 crea una malla de 10×10 (100 cuadrantes).

2.2. Cámara

La segunda pestaña **Cámara** controla los aspectos de la captura de video. Se divide en tres secciones:

2.2.1. Configuración Básica

Presenta cuatro campos numéricos:

- **Índice de Cámara**: Rango de 0 a 5, selecciona la cámara conectada.
- **Ancho (px)**: Resolución horizontal entre 640 y 1920.
- **Alto (px)**: Resolución vertical entre 480 y 1080.
- **FPS Objetivo**: Entre 15 y 60.

2.2.2. Resoluciones Rápidas

Tres botones predefinidos:

- **1080p**: 1920×1080 px.
- **720p**: 1280×720 px.
- **480p**: 640×480 px.

2.2.3. Compensación de Delay

- **Delay Cámara (s)**: Campo numérico con incrementos de 0.001 segundos, rango entre 0.05 y 0.5 segundos. Permite compensar el tiempo que transcurre desde que el robot se mueve hasta que la cámara captura esa nueva posición. El valor predeterminado de 0.15 segundos es adecuado para la mayoría de configuraciones.
- **Horizonte Predicción**: Campo que acepta valores enteros entre 1 y 10. Indica cuántos cuadros (frames) hacia el futuro debe predecir el sistema la posición del robot. Un horizonte de 3 significa que el sistema anticipa la posición tres frames adelante, compensando así el retardo del sistema completo.

2.3. Control

La tercera pestaña **Control** contiene parámetros de navegación, dividida en tres secciones:

2.3.1. Ganancias Proporcionales (Kp)

- **Kp Lineal**: Campo decimal entre 1.0 y 200.0, con valor predeterminado de 80.0. Determina qué tan fuerte es la respuesta del robot cuando está lejos de su objetivo. Valores altos hacen que el robot se mueva más rápido hacia el objetivo, pero pueden causar movimientos bruscos u oscilaciones. Valores bajos producen movimientos suaves pero lentos.
- **Kp Angular (Movimiento)**: Rango de 1.0 a 300.0, predeterminado en 120.0. Controla qué tan agresivamente el robot ajusta su orientación para apuntar hacia la dirección de movimiento mientras se desplaza. Valores altos hacen que el robot gire rápidamente para alinearse, valores bajos permiten curvas más suaves.
- **Kp Angular (Fijo)**: También entre 1.0 y 300.0, con valor inicial de 120.0. Se utiliza cuando el modo de orientación automática está desactivado, controlando qué tan firmemente el robot mantiene una orientación fija mientras se mueve. Útil para aplicaciones donde se requiere que el robot mantenga una dirección específica independientemente de hacia dónde se mueve.

2.3.2. Ganancias Integrales (Ki)

- **Ki Lineal**: Campo entre 0.0 y 50.0, predeterminado en 15.0. Acumula el error de posición a lo largo del tiempo para corregir desviaciones causadas por fricciones, desniveles del piso, o imprecisiones del sistema. Un valor de 0 desactiva esta corrección.
- **Ki Angular**: Rango de 0.0 a 50.0, con valor inicial de 20.0. Similar al Ki Lineal pero para la orientación del robot. Corrige desviaciones angulares persistentes.
- **Max Integral Lineal**: Límite entre 10 y 200, predeterminado en 50.0. Evita que la acumulación de error lineal crezca sin control, lo cual podría causar sobrepasamiento excesivo del objetivo. Es una protección de seguridad del sistema.
- **Max Integral Angular**: Entre 10 y 100, valor inicial de 30.0. Limita la acumulación de error angular por las mismas razones de estabilidad.
- **Botón Reset Integradores**: Permite reiniciar manualmente todos los acumuladores de error a cero. Útil si el robot muestra comportamiento errático debido a acumulación excesiva de error, o al cambiar significativamente los parámetros de control.

2.3.3. Límites y Tolerancias

- **Vel. Max Lineal**: Campo entre 50 y 500, predeterminado en 250. Establece la velocidad máxima permitida para el movimiento de traslación del robot. Funciona como límite de seguridad para evitar comandos excesivos que podrían dañar el robot o causar movimientos peligrosos.
- **Vel. Max Angular**: Rango de 50 a 500, valor inicial de 250. Limita la velocidad máxima de rotación del robot. Previene giros demasiado bruscos que podrían desestabilizar al robot o causar derrapes.
- **Tolerancia Posición**: Valor decimal entre 0.1 y 1.0, predeterminado en 0.20. Define qué tan cerca debe estar el robot del objetivo para considerarlo alcanzado. Una tolerancia de 0.20 en una malla de 10×10 equivale aproximadamente a 2 del tamaño total del espacio. Valores muy pequeños aumentan la precisión pero pueden causar que el robot oscile sin detenerse completamente.

2.4. Firebase

La cuarta pestaña **Firebase** gestiona conexión remota. Dos secciones:

2.4.1. Configuración de Conexión

- **URL de Firebase**: Dirección completa de la base de datos.
- **Ruta del Robot**: Ubicación jerárquica dentro de la base.
- **Archivo de Credenciales**: Ruta del JSON de credenciales.

2.4.2. Estado de Conexión

- **Indicador de Estado**: Conectado, Desconectado o Error.
- **Botón Conectar**: Establece conexión.
- **Botón Desconectar**: Cierra conexión.

2.5. Manual

La quinta pestaña **Manual** permite control directo.

2.5.1. Control de Velocidad

- **Deslizador de Velocidad**: Entre 50 y 250.

2.5.2. Control de Dirección

- ↑: Avanza.
- ←: Izquierda.
- →: Derecha.
- ↓: Retrocede.

2.5.3. Controles de Rotación y Emergencia

- **Girar Izq**.
- **Girar Der**.
- **Parar**.

2.6. Avanzado

La sexta pestaña **Avanzado** contiene configuraciones especializadas.

2.6.1. Seguimiento Robusto

- **Max Frames Perdidos**: Campo numérico entre 5 y 50, con valor predeterminado de 15. Define cuántos cuadros consecutivos puede el sistema no detectar el robot antes de considerar que se perdió completamente y detener los movimientos. Un valor de 15 significa que con 30 FPS, el sistema tolera hasta 0.5 segundos de pérdida de detección antes de detenerse. Valores altos son más tolerantes a oclusiones temporales pero pueden permitir que el robot continúe moviéndose sin visión por más tiempo.
- **Casilla Habilitar Predicción**: Checkbox (casilla de verificación) que activa o desactiva el sistema de predicción de posición. Cuando está marcada, si el robot temporalmente sale del campo de visión o no es detectado claramente, el sistema intenta predecir su posición basándose en su trayectoria reciente y continúa enviando comandos. Cuando está desmarcada, cualquier pérdida de detección detiene inmediatamente el control..
- **Umbral de Calidad**: Deslizador entre 10.0 y 100.0, predeterminado en 30.0. Establece el nivel mínimo de nitidez que debe tener la imagen para considerar válida la detección de marcadores ArUco. Valores bajos aceptan imágenes borrosas o de baja calidad (útil en condiciones de iluminación pobre), mientras que valores altos exigen imágenes muy nítidas (reduce falsos positivos pero puede rechazar detecciones válidas en condiciones no ideales).

2.6.2. Estado de Seguimiento

- **Estado**: Detectado, Perdido, Inactivo.

- **Frames Perdidos**: Contador numérico que muestra cuántos cuadros consecutivos han pasado sin detectar el robot. Se actualiza constantemente durante la operación. Cuando alcanza el valor de "Max Frames Perdidos", el robot se detiene automáticamente.
- **Predicción**: Activa, Inactiva, Deshabilitado.

Capítulo 3: Acomodo del escenario

Previo a la ejecución de cualquier prueba de movimiento con el carrito, es indispensable realizar una correcta preparación del entorno de trabajo. Esta etapa es fundamental para asegurar condiciones óptimas de operación, reducir errores de detección y garantizar la correcta interpretación espacial por parte del sistema de visión. A partir de lo aprendido en la Práctica 01, se cuenta con una estructura de perfiles de aluminio tipo Bosch, sobre la cual se monta la cámara encargada de capturar el entorno. Esta estructura debe ubicarse en una posición central y estable dentro del área de trabajo, permitiendo una vista amplia y sin obstrucciones del espacio en el que operará el carrito.

Una vez instalada la estructura, se procede a despejar completamente el área de obstáculos que puedan interferir con la visibilidad de la cámara o la movilidad del robot. A continuación, se deben colocar los códigos ArUco en las cuatro esquinas del área de trabajo. Estos marcadores actúan como referencias espaciales para la calibración y localización del robot dentro del entorno.

Es importante que cada código ArUco se oriente correctamente, alineando sus ejes X e Y de manera coherente entre sí, siguiendo un mismo sistema de coordenadas. Esta alineación es crucial para garantizar que la interpretación de la posición y orientación del carrito sea precisa durante su operación.

Una vez posicionados todos los códigos, se debe realizar una verificación visual utilizando la cámara conectada al sistema. Es recomendable utilizar un software de visualización en tiempo real (por ejemplo, una interfaz ROS o una aplicación de captura de imagen) para confirmar que todos los códigos sean detectados correctamente y que no existan zonas fuera del campo de visión de la cámara. Puedes tomar como referencia la siguiente imagen:

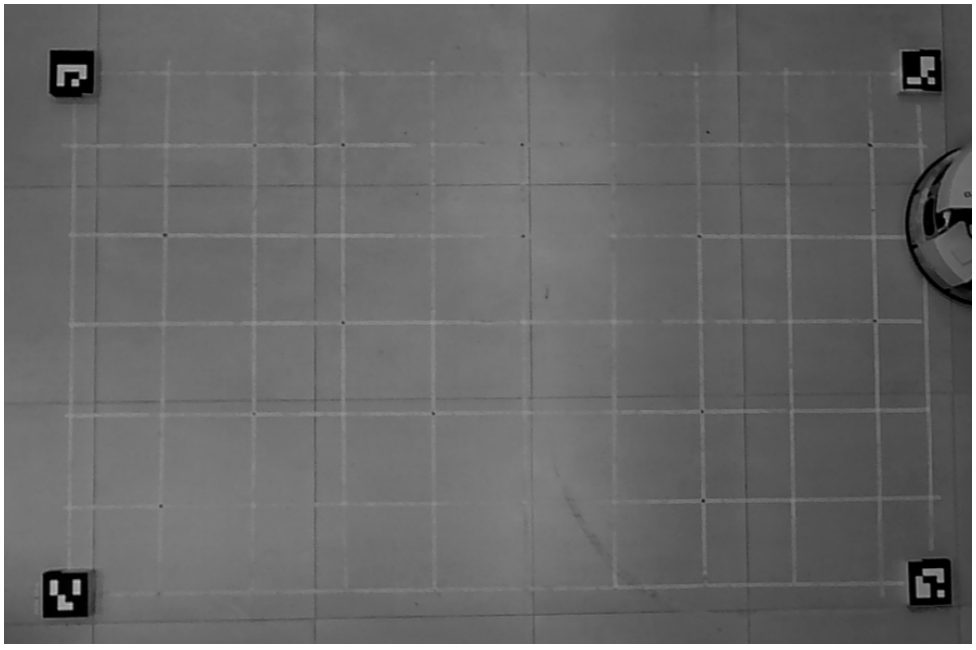

Figura 2: Códigos en escenario.

Capítulo 4: Iniciar Cámara y Conectar al Carrito

Una vez que el escenario ha sido correctamente acondicionado y el carrito ha sido montado sobre el área de trabajo, se puede proceder con las primeras pruebas del sistema. Para ello, es necesario configurar adecuadamente dos componentes principales: la cámara de visión y el sistema de base de datos.

4.1 Cámara

Primero nos enfocaremos en la configuración inicial de la cámara, la cual es esencial para la detección de los marcadores ArUco y la correcta interpretación espacial del entorno.

4.1.1 Conexión y reconocimiento de la cámara

El primer paso consiste en asegurar que la cámara esté correctamente conectada a la computadora mediante un puerto USB. A continuación, se debe iniciar la interfaz gráfica del sistema. En la sección correspondiente a la cámara, se debe establecer el parámetro del índice del dispositivo de captura. Este valor depende de la configuración del equipo:

- Si la computadora cuenta con una cámara integrada y además se ha conectado una cámara USB, el índice correspondiente a la cámara USB suele ser 1.
- Si únicamente se dispone de la cámara USB (es decir, no hay cámara integrada), el índice adecuado será generalmente 0.

Seleccionar el índice correcto es crucial para que el sistema identifique el dispositivo desde el cual debe capturar las imágenes.

4.1.2 Configuración de resolución y parámetros de captura

Una vez establecido el índice, se procede a configurar la resolución de captura. Esta puede establecerse de forma manual ingresando los valores deseados de ancho (*width*) y alto (*height*), o bien seleccionarse mediante botones de resolución preestablecida que ofrece la interfaz.
Posteriormente, se debe definir la tasa de cuadros por segundo (FPS) deseada para la captura de video. Este valor impacta directamente en la fluidez de la detección y debe ajustarse conforme a las capacidades de la cámara y el rendimiento del sistema.

4.1.3 Ajuste de compensación por retardo (delay)

Otro parámetro a considerar es la compensación por retardo (*delay compensation*), el cual se encarga de sincronizar correctamente la información visual con el sistema de control del carrito. Por defecto, estos valores vienen preconfigurados, pero pueden ajustarse de manera personalizada utilizando los conceptos abordados en el Capítulo 2.

4.1.4 Verificación del funcionamiento del sistema

Una vez completada la configuración de todos los parámetros anteriores, el sistema está listo para ser iniciado. En la interfaz, se debe presionar el botón **Iniciar sistema**, ubicado en la parte inferior izquierda. Si el proceso se ejecuta correctamente, se abrirá una ventana emergente indicando que la cámara ha sido activada con éxito (dependiendo de la resolución seleccionada). En caso de existir un error —por ejemplo, si la cámara no fue detectada o está en uso por otro proceso— se mostrará un mensaje de advertencia correspondiente.
Si la cámara fue configurada adecuadamente, se podrá observar en la sección de vista (ubicada al lado izquierdo de la interfaz) el flujo de video en tiempo real, mostrando el entorno de trabajo

con los códigos ArUco visibles y correctamente iluminados. Esta visualización constituye una validación preliminar de que el sistema de visión está listo para iniciar las pruebas experimentales con el carrito.

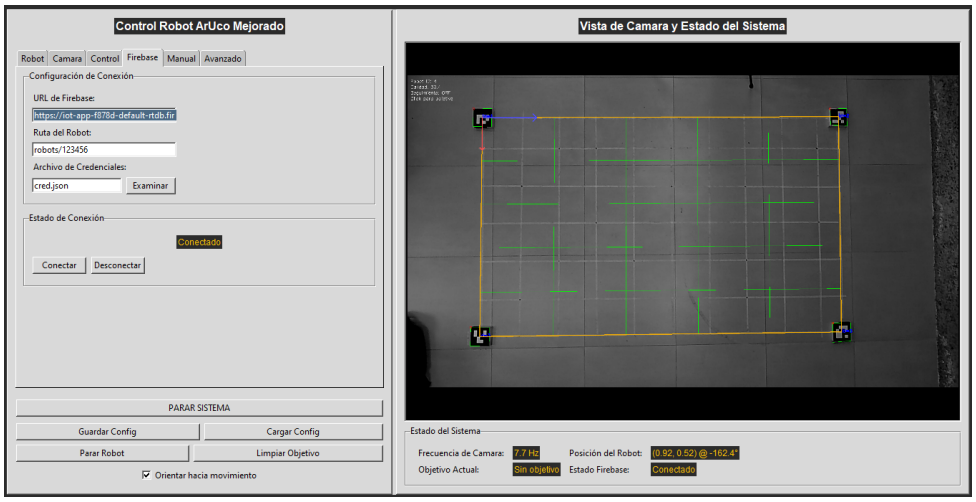

Figura 3: Cámara encendida.

4.2 Firebase

Tras la correcta configuración de la cámara, el siguiente paso fundamental en el sistema es establecer la conexión con la base de datos remota, desde donde se obtendrán o almacenarán los datos relevantes para el control y monitoreo del carrito. Este sistema utiliza una base de datos en tiempo real proporcionada por Firebase, ya preconfigurada en el entorno de desarrollo.

4.2.1 Requisitos previos

Para establecer la conexión, es importante contar con los siguientes elementos:

- La ruta asignada al robot, definida previamente en el Capítulo 0.
- El archivo de credenciales `cred.json`, incluido en el repositorio oficial del proyecto en GitHub.

Estos elementos son necesarios para autenticar correctamente el sistema ante la base de datos y ubicar al robot dentro de la estructura jerárquica del servidor.

4.2.2 Parámetros de conexión

La interfaz de configuración de base de datos cuenta con varios campos preestablecidos. En la mayoría de los casos, no se recomienda modificar estos parámetros, salvo que se requiera migrar a una nueva instancia de base de datos (situación no contemplada en esta práctica).
Los parámetros clave son los siguientes:

- **URL de la base de datos**: El sistema utiliza como endpoint el siguiente URL de Firebase: `https://iot-app-f878d-default-rtdb.firebaseio.com/`. Este valor ya viene cargado por defecto en la interfaz y no debe modificarse.
- **Ruta del robot**: Por defecto, esta ruta se establece como `robots/123456`. Sin embargo, si se realizó una modificación en el código del carrito durante el desarrollo (por ejemplo, cambiando el identificador del robot), es imprescindible actualizar también esta ruta en la interfaz para mantener la coherencia entre el dispositivo físico y la base de datos.

- **Archivo de credenciales (cred.json)**: Este archivo contiene las claves necesarias para autenticar el acceso a Firebase. Su ubicación está predefinida y solo debe cambiarse si el archivo fue movido de lugar o reemplazado (lo cual tampoco se recomienda durante esta etapa de pruebas).

4.2.3 Establecimiento de la conexión

Una vez verificados y configurados los parámetros mencionados, se debe presionar el botón **Conectar** en la interfaz. Si la conexión se realiza exitosamente, el sistema mostrará una ventana emergente de confirmación, y el estado de conexión cambiará de *Desconectado* a *Conectado*. Este cambio de estado indica que el sistema ya puede intercambiar datos en tiempo real con la base de datos, lo cual habilita funcionalidades como la lectura de instrucciones para el carrito o la escritura de datos de posición y estado.

Capítulo 5: Movimiento por Selección de Objetivo

Una vez que se ha configurado correctamente el sistema y el carrito está listo en el escenario, podemos proceder con las primeras pruebas de movimiento por selección de objetivo. En esta fase, el objetivo es que el carrito sea capaz de desplazarse hacia un punto determinado dentro del escenario mediante la selección de dicho objetivo en la interfaz.

5.1 Preparación del Escenario

El primer paso consiste en colocar el carrito en el centro de la malla dentro del escenario. Esto nos permitirá tener una visión clara de su posición inicial y su relación con el entorno, lo cual es crucial para evaluar su desempeño al moverse. Una vez que el carrito se encuentra en su posición de inicio, debemos ser capaces de visualizar tanto la malla del escenario como la posición del carrito en la sección de vista. Esta configuración es esencial para que el sistema reconozca el punto de partida y la zona en la que debe operar.

Figura 4: Escenario con malla visible y carrito en posición inicial.

Además, podemos ajustar el número de cuadrantes del escenario según los parámetros definidos en el Capítulo 2. Este ajuste se realiza en la pestaña de cámara, lo que nos permite modificar la resolución del área de trabajo para mejor adaptación al entorno físico.

5.2 Establecimiento del Objetivo

Una vez que el escenario está correctamente configurado y que el carrito es visible, podemos proceder a establecer el objetivo de movimiento. Para ello, basta con hacer un clic izquierdo en el lugar dentro de la malla donde deseamos que el carrito se desplace. Al hacer clic, en la parte inferior de la ventana de vista, se deberá mostrar el objetivo seleccionado, así como una línea amarilla que conecta la posición actual del carrito con el objetivo.

Figura 5: Selección de objetivo y línea de movimiento.

Este proceso de selección es simple, pero esencial, ya que define la trayectoria que el carrito debe seguir. Una vez que el objetivo está establecido, el carrito debería comenzar a moverse automáticamente hacia esa ubicación, ajustando su rumbo de acuerdo a los parámetros de control que hemos definido.

5.3 Movimiento del Carrito

El movimiento del carrito hacia el objetivo es controlado por un sistema de retroalimentación basado en el error entre la posición actual y la posición deseada. Es fundamental observar el comportamiento del carrito durante este movimiento para asegurarnos de que se dirige correctamente hacia el objetivo sin desviaciones. El sistema de control emplea una estrategia de ajuste de los parámetros de control proporcional-integral (PI), lo cual nos permite modificar la velocidad y la precisión del movimiento.

Aunque los valores iniciales del PI vienen preconfigurados, es posible ajustarlos según los resultados observados durante las pruebas. Para realizar estos ajustes, debemos referirnos a los rangos de valores establecidos en el Capítulo 2, donde se describen los parámetros de control más adecuados para optimizar el rendimiento del sistema.

5.4 Ajuste de Parámetros de Control

Es posible que durante las pruebas de movimiento, el carrito no llegue exactamente al objetivo debido a diversos factores, como la calibración del sistema o las características del entorno. En este caso, es necesario ajustar los parámetros de control PI para mejorar la precisión y estabilidad del movimiento.

El ajuste de estos parámetros puede realizarse directamente desde la pestaña de control , donde se visualizan los valores actuales y se permite modificar los coeficientes proporcional e integral.

Para obtener el mejor desempeño, es recomendable realizar una serie de pruebas con diferentes valores hasta encontrar los que ofrezcan un movimiento más fluido y preciso.

Es importante tener en cuenta que el ajuste de los parámetros debe hacerse de manera gradual. A medida que se realicen las pruebas, es probable que el carrito comience a moverse de manera más precisa y eficiente, lo que nos permitirá comprobar la efectividad de los cambios realizados.

5.5 Conclusiones del Movimiento por Selección de Objetivo

Al finalizar las pruebas de movimiento por selección de objetivo, deberíamos ser capaces de observar un comportamiento adecuado del carrito, que se mueva de manera controlada y eficiente hacia el objetivo seleccionado. Además, es fundamental que el sistema permita realizar ajustes en tiempo real a los parámetros de control, de forma que podamos optimizar continuamente el rendimiento del sistema.

En resumen, el movimiento por selección de objetivo es una de las funcionalidades clave del sistema, ya que permite al carrito operar de manera autónoma dentro del entorno, respondiendo a las solicitudes de desplazamiento de manera precisa. Sin embargo, es necesario seguir afinando los controles para lograr un movimiento aún más robusto y eficiente en diferentes condiciones de operación.

Capítulo 6: Conclusiones y Desafíos Futuros

6.1 Conclusiones del Aprendizaje

Al finalizar esta práctica, se han integrado con éxito los tres pilares de un sistema de control de visión: la percepción visual, la comunicación remota, y el control de movimiento.

- Integración de Subsistemas: Se ha comprobado que la arquitectura basada en dos componentes (Interfaz Gráfica en PC y Código del Carrito en Raspberry Pi) se comunica de manera efectiva en tiempo real a través de Firebase. Esta conexión garantiza la trazabilidad y el control remoto del sistema.
- Percepción Espacial: La correcta configuración de la cámara y la ubicación de los marcadores ArUco permitieron al sistema de visión ubicar con precisión el carrito y el entorno de trabajo (la malla) en la Sección de Vista.
- Control Autónomo: La prueba de Movimiento por Selección de Objetivo confirmó la funcionalidad del controlador Proporcional-Integral (PI). El sistema es capaz de calcular el error entre la posición actual y el objetivo, y generar comandos de velocidad lineal y angular para guiar al carrito, demostrando un lazo cerrado de control funcional.
- Optimización de Parámetros: Se comprendió la función crítica de las Ganancias Proporcionales (Kp) y las Ganancias Integrales (Ki). El ajuste fino de estos valores es indispensable para lograr que el carrito se mueva de manera eficiente (rápida, con alta Kp) y precisa (minimizando el error residual con Ki), sin generar movimientos bruscos u oscilaciones.

6.2 Desafíos y Próximos Pasos

Para profundizar en el entendimiento del sistema y prepararse para desafíos de navegación más avanzados, se proponen los siguientes puntos de exploración:

6.2.1. Exploración del Control Robusto

El verdadero desafío en robótica móvil es la robustez ante condiciones no ideales.

- Puesta a Prueba de la Predicción: En la pestaña Avanzado, active la casilla Habilitar Predicción. Luego, pruebe a tapar brevemente la cámara o el marcador ArUco del carrito. Observe cómo el sistema maneja la pérdida temporal de detección al intentar predecir su posición basándose en la trayectoria reciente, evitando la detención inmediata si la pérdida es corta.
- Límites de Calidad: Modifique el Umbral de Calidad.
  - *Valor bajo (e.g., 10.0):* Permite la detección incluso en imágenes borrosas o con iluminación deficiente.
  - *Valor alto (e.g., 80.0):* Exige una imagen muy nítida, lo que puede aumentar la fiabilidad, pero hace que el sistema sea más sensible a las variaciones de iluminación.

6.2.2. Ampliación del Control y Navegación

El movimiento por objetivo único es la base; los siguientes pasos buscan trayectorias más complejas.

- Coherencia de Identificación: Asegúrese de haber personalizado el ROBOT_ID_PATH en el script de la Raspberry Pi (`carrito.py`) y en la pestaña Firebase de la interfaz. Comprender que esta ruta es la clave de comunicación única en la nube.
- Secuencias de Movimiento: ¿Cómo podría el sistema moverse a tres objetivos diferentes de manera secuencial? Este ejercicio mental sienta las bases para el concepto de *Planificación de Rutas (Path Planning)*, donde el robot recibe una lista de coordenadas a seguir, y no solo una.
- Control por Velocidad: Utilice la pestaña Manual para enviar comandos directos de movimiento (adelante, atrás, girar) sin la retroalimentación de la cámara. Esto ayuda a diferenciar el Control en Lazo Abierto (control manual directo) del Control en Lazo Cerrado (control PI con visión) y a sentir la respuesta pura del hardware del carrito.