Manual de Operación y Configuración del Duckiebot (Ente)

Elaborado por: Steven Ignacio Martinez Jara
Revisado para: Laboratorio Duckietown

29 de enero de 2026

Índice

1. Introducción General

El propósito de este manual es servir como una guía centralizada para el ensamblaje, configuración, operación y mantenimiento de los robots Duckiebot utilizados en el laboratorio. Este documento está diseñado para estandarizar los procedimientos y proporcionar una referencia técnica clara para todos los usuarios, desde estudiantes que se inician en la plataforma hasta investigadores avanzados.

El manual cubrirá los siguientes aspectos clave:

- Descripción detallada de los componentes de hardware.
- Procedimientos de ensamblaje.
- Guía de configuración inicial del software y conexión a la red.
- Protocolos de operación y seguridad.
- Diagnóstico y solución de problemas comunes.

2. Componente: Fuente de Alimentación (Duckiebattery)

La fuente de alimentación es un componente crítico que garantiza la autonomía y el correcto funcionamiento del Duckiebot. El modelo estándar es la Duckiebattery (DB-C-DBatt).

2.1. Descripción General de la Duckiebattery

La Duckiebattery es una fuente de alimentación inteligente diseñada específicamente para las exigencias de aplicaciones robóticas móviles. Sus características principales incluyen la capacidad de monitorear el estado de carga, permitir un apagado seguro del sistema a través de software y garantizar una alimentación ininterrumpida a la unidad de cómputo durante los ciclos de carga.

2.2. Especificaciones Técnicas

- **Capacidad:** 10Ah a 3.7V.
- **Entrada de Carga:** Micro USB, 5V hasta 2A.
- **Salida de Alimentación:** 2 x USB tipo A, 5V hasta 4A (combinado). Máximo 2.5A en un solo puerto.
- **Tiempo de Carga (aprox.):** 5 horas para 0-100 % y 4 horas para 0-90 % con un adaptador de 2A.
- **Peso:** 189g (con carga completa).

Figura 1: La Duckiebattery, fuente de alimentación de 5V diseñada para robótica.

2.3. Precauciones de Seguridad y Manejo

Las baterías de iones de litio (Li-Ion) requieren un manejo cuidadoso para prevenir daños al equipo y garantizar la seguridad del personal. Es imperativo seguir las siguientes directrices.

2.3.1. Prácticas Recomendadas

- **Inspección Visual:** Verifique que la carcasa de la batería no presente hinchazón, fisuras o daños.
- **Almacenamiento:** Guarde la batería en un área fresca, seca y ventilada.
- **Disposición:** Deseche de forma inmediata y segura cualquier batería que presente daños evidentes.

2.3.2. Prácticas Prohibidas

- **NO** conecte una fuente de carga con un voltaje superior a 5V.
- **NO** conecte una fuente de voltaje externa a los puertos de salida USB.
- **NO** intente abrir, perforar, aplastar o incinerar la batería.
- **NO** sumerja la batería en ningún tipo de líquido.

2.4. Operación de la Batería

2.4.1. Indicadores LED y Proceso de Carga

La Duckiebattery está equipada con cinco LEDs que indican el nivel de carga. Para visualizar el estado, presione una vez el botón lateral. Para cargar, conecte un adaptador de 5V/2A al puerto Micro USB. Los LEDs parpadearán (1 Hz) durante la carga.

Nota Importante: Si la batería está en descarga profunda, es posible que los LEDs no respondan durante los primeros 30 minutos de carga. Esto es normal.

2.4.2. Estados de Operación y Transiciones

La batería opera según un diagrama de estados definido que gestiona la carga y la salida de energía. Los estados principales son: *Sleep Mode*, *Idle Mode*, *Active Mode* y *Boot Mode*. Las transiciones entre estos estados se controlan mediante el botón lateral y la detección de carga o consumo.

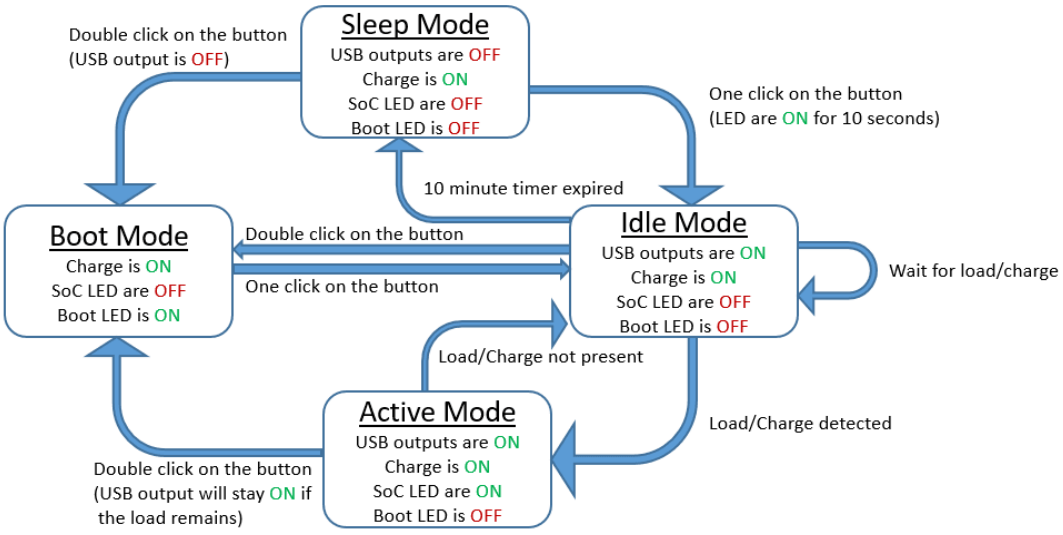

Figura 2: Diagrama de estados de la Duckiebattery.

2.4.3. Descripción de los Puertos de Salida USB

- **USB OUT-1 (Actuadores):** Destinado a motores y LEDs.
- **USB OUT-2 (Unidad de Cómputo):** Proporciona una salida de 5V ininterrumpida. Conectar aquí la Raspberry Pi o Jetson Nano.

2.5. Diagnóstico y Solución de Problemas de la Batería

La falla más común no reside en la batería, sino en las conexiones. Utilice cables USB de alta calidad y de una longitud no superior a 30 cm.

2.5.1. La batería no parece aceptar carga

- **Causa posible:** La batería se encuentra en modo de protección. *Resolución:* Conecte la unidad a un cargador y espere más de 30 minutos.
- **Causa posible:** La batería está sobrecalentada. *Resolución:* Desconecte y deje reposar durante 1 hora.

2.5.2. Uno o ambos puertos USB no suministran energía

- **Causa posible:** La batería no está en modo inactivo. *Resolución:* Presione el botón lateral una vez.
- **Causa posible:** El puerto entró en modo de protección por sobrecorriente. *Resolución:* Desconecte las cargas y deje reposar la unidad durante 30 minutos.

3. Preparación y Flasheo de la Tarjeta SD

Este proceso instala el sistema operativo Duckiebot OS en la tarjeta microSD, junto con la configuración inicial de red y del robot. Es el primer paso fundamental para darle vida a tu Duckiebot.

3.1. Flasheo mediante Línea de Comandos (CLI)

El método recomendado utiliza la herramienta `dts` para inicializar la tarjeta de forma guiada y segura.

Paso 1: **Conectar la Tarjeta SD.** Inserta la tarjeta microSD en un lector y conéctala a tu computadora.

Paso 2: **Ejecutar el Comando de Inicialización.** Abre una terminal y ejecuta el siguiente comando, reemplazando los valores correspondientes.

```
dts init_sd_card --hostname [nombre_del_duckiebot] --configuration [modelo]
```

- `-hostname`: Es el nombre que le darás a tu robot en la red.
- `-configuration`: Es el modelo específico de tu Duckiebot. Es muy importante que uses el correcto:
  - **DB21M** para el Jetson Nano de **2GB**.
  - **DB21J** para el Jetson Nano de **4GB**.

Paso 3: **Configuración de la Red Wi-Fi.** Para que el robot se conecte a tu red, debes añadir el parámetro `-wifi`. Puedes añadir varias redes separándolas por comas (sin espacios).

```
dts init_sd_card ... --wifi mi_red:mi_clave,otra_red:otra_clave
```

El formato es siempre `nombre_de_la_red:contraseña`.

Paso 4: **Seguir las Instrucciones en Pantalla.** El programa te guiará para que selecciones el dispositivo correcto correspondiente a tu tarjeta SD y comenzará el proceso de flasheo. Al finalizar, te indicará que puedes retirar la tarjeta.

Solución de Problemas Comunes

- **El flasheo falla con un error de `Bad archive`:** Esto significa que la descarga de la imagen del sistema está corrupta. Añade la opción `-no-cache` a tu comando `init_sd_card` para forzar una nueva descarga.
- **La tarjeta no se escribe o el proceso es muy rápido:** Verifica que el seguro de protección contra escritura de tu adaptador de SD no esté activado.

3.2. Verificación y Pruebas Iniciales

Una vez que la tarjeta SD esté flasheada, insértala en tu Duckiebot y enciéndelo. Después de unos minutos, el robot debería conectarse a la red Wi-Fi que configuraste.

El siguiente paso es acceder al **Dashboard** de tu robot desde un navegador web en la dirección:

```
http://[nombre_del_duckiebot].local
```

Desde el Dashboard podrás:

- **Verificar el estado de los componentes:** Comprueba que la cámara, los motores y los sensores sean detectados correctamente.
- **Realizar pruebas de movimiento:** Utiliza las herramientas del dashboard para probar que los motores responden.
- **Observar la vista de la cámara:** Confirma que el flujo de video se está recibiendo.

4. Protocolos de Operación (Modelo DB21)

Esta sección describe los procedimientos estándar para la operación diaria de un Duckiebot modelo DB21.

4.1. Requisitos Previos

- Un robot DB21 completamente ensamblado.
- Un Duckiebot DB21 inicializado con la versión de firmware 1.2.2 o superior.

4.2. Procedimiento de Encendido

Para encender un robot DB21, presione el botón lateral de la batería una vez. Se observará la siguiente secuencia de arranque:

1. Los LEDs del Duckiebot y de la computadora a bordo se encienden.
2. Tras unos segundos, el dongle Wi-Fi comienza a parpadear.
3. Los LEDs del robot cambian a un color blanco fijo.
4. Finalmente, la pantalla superior y el botón se activan.

4.3. Procedimiento de Apagado

Advertencia: El protocolo de apagado correcto requiere que la Duckiebattery tenga la versión de software 2.0.0 o superior. Antes de apagar, verifique que el proceso de arranque del robot ha finalizado (estado R̈eadyäl ejecutar `dts fleet discover`).

Existen tres métodos recomendados para apagar un DB21:

- **Usando el botón superior (método preferido):** Mantenga presionado el botón superior (no el de la batería) durante 5 segundos y suéltelo. El botón parpadeará, los LEDs del robot se apagarán y en ∼10 segundos, la computadora a bordo y el ventilador se apagarán.
- **Usando el comando dts:** Ejecute en la terminal de su computador:
  `dts duckiebot shutdown <ROBOT_NAME>`
- **A través del Dashboard:** Navegue a `http://<ROBOT_NAME>.local` en un navegador web y seleccione la opción "Shutdown.[en] el menú de encendido.

Apagado Forzado (Hard Shutdown): Este método solo debe usarse si los anteriores fallan, ya que podría corromper el software. Como último recurso, desconecte el cable microUSB del puerto marcado como `5Vraspi` en el HUT.

4.4. Proceso de Carga

1. Conecte el cable de carga al puerto microUSB libre en el HUT (placa superior).

   Nota: Para minimizar el estrés mecánico, se recomienda dejar el cable conectado al HUT y utilizar el extremo USB-A para conectar/desconectar de la fuente de alimentación.

2. Conecte el cargador a una fuente de 5V y 2A.

   Nota: Un amperaje superior no causará problemas, pero un voltaje incorrecto activará el modo de protección de la batería.

Indicadores de carga: Si el robot está encendido, un ícono de batería aparecerá en la pantalla. Si está apagado, los LEDs de la batería se iluminarán. En ambos casos, un LED verde en el HUT indicará la entrada de energía.

4.5. Acceso Remoto por SSH

Para acceder a la terminal del robot de forma remota, utilice el protocolo SSH con el siguiente comando:

`ssh duckie@<HOSTNAME>.local`

La contraseña por defecto es `quackquack`.

4.6. Actualización de Firmware

- **Actualización de la Duckiebattery:** Consulte la documentación oficial para el procedimiento de actualización del firmware de la batería.
- **Actualización del HUT:** Este procedimiento rara vez es necesario. Consulte la documentación oficial si se requiere.

4.7. Diagnóstico y Solución de Problemas de Operación

- **Síntoma:** El botón de encendido superior no apaga el Duckiebot.

  Resolución: Asegúrese de mantener presionado el botón durante 5 segundos. Si persiste, ejecute `dts duckiebot update <your_robot>` y luego `dts duckiebot reboot <your_robot>`.

- **Síntoma:** El Duckiebot está atascado en un ciclo de reinicios con batería baja.

  Resolución: Desconecte todos los cables del HUT excepto el de carga. Permita que la batería se cargue durante al menos 5 horas antes de reconectar todo.

4.8. Casos de Uso Frecuentes y Soluciones Específicas del Laboratorio

Esta sección documenta problemas específicos encontrados durante la operación de los Duckiebots en el laboratorio y proporciona los procedimientos de solución estandarizados.

4.8.1. Añadir una Nueva Red Wi-Fi (Post-Flasheo)

- **Síntoma:** Se necesita operar un Duckiebot en una nueva ubicación (e.g., en casa, en otro laboratorio) con una red Wi-Fi diferente a la configurada inicialmente.
- **Causa:** El robot solo conoce las credenciales de las redes que se le proporcionaron durante el flasheo. Es necesario añadir la nueva red a su lista de conexiones conocidas.
- **Solución:** Se debe editar el archivo de configuración de red dentro del robot. Existen tres métodos para acceder a este archivo, ordenados de menor a mayor complejidad.

  Fase 1: Acceder y Configurar la Nueva Red Método A: En Vivo por Wi-Fi (Si ya tienes acceso)
  Úsalo si el robot está conectado a una red antigua (ej. laboratorio) y quieres añadir la de tu casa antes de llevártelo.

1. Conéctate por SSH al robot en la red actual: `ssh duckie@myduckiebot01.local`
2. Ve al paso 3 del Método B para editar el archivo.

Método B: Conexión Directa por Cable Ethernet (Recomendado)
Úsalo si el robot no se conecta al Wi-Fi y no quieres desarmarlo.

1. **Conexión Física:** Conecta un cable Ethernet directamente entre el puerto RJ45 del Duckiebot (en la Jetson Nano) y tu computadora (o un router cercano).
2. **Acceso SSH:** Espera unos segundos y conecta por SSH. La dirección `.local` funciona a través del cable gracias al protocolo mDNS.

   ```
   ssh duckie@myduckiebot01.local
   ```

3. **Editar configuración:** Abre el archivo de credenciales Wi-Fi con permisos de administrador.

   ```
   sudo nano /etc/wpa_supplicant.conf
   ```

4. **Añadir la red:** Ve al final del archivo y agrega un nuevo bloque `network={}`. **¡Cuidado con las mayúsculas y comillas!**

   ```
   # Nueva Red
   network={
       ssid="NOMBRE_EXACTO_WIFI"
       psk="Tu_Contrasena"
       key_mgmt=WPA-PSK
       priority=1
   }
   ```

5. **Guardar y Probar:** Guarda con `Ctrl+O`, Sal con `Ctrl+X` y reinicia con `sudo reboot`. Desconecta el cable Ethernet mientras se reinicia.

Método C: Editando la Tarjeta SD (Último Recurso)
Usa este método solo si no tienes cable Ethernet o el sistema no arranca.

1. **Apagar y Retirar:** Apaga el robot y retira la tarjeta microSD.
2. **Conectar a PC:** Inserta la tarjeta en tu computadora.
3. **Acceder a la Partición:** Monta la partición llamada `rootfs` (o `APP`).
4. **Editar:** Edita el archivo en la ruta `/etc/wpa_supplicant.conf` añadiendo el bloque `network` descrito arriba.
5. **Finalizar:** Guarda, expulsa la tarjeta, insértala en el robot y enciende.

Fase 2: Encontrar y Conectar al Robot en la Nueva Red Una vez que el robot se conecte al nuevo Wi-Fi, es posible que su dirección IP cambie.

1. **Limpiar IP antigua (si usas hosts):** Si configuraste IPs estáticas en `/etc/hosts`, comenta la línea antigua añadiendo un `#` al inicio.

2. **Descubrir la nueva IP:** Haz ping al nombre del robot.

   ```
   ping myduckiebot01.local
   ```

   La respuesta te revelará la nueva dirección: `PING ... (192.168.1.22)`.

3. **Verificar conexión:** Si el ping responde correctamente, ya tienes control total sobre el robot en la nueva red.

Información Adicional y Verificación de Internet Para verificar que tu Duckiebot tiene salida a internet (necesaria para descargar actualizaciones o Docker), ejecuta:

```
sudo curl google.com
```

Si recibes código HTML como respuesta, la conexión es exitosa.

Diagnóstico Avanzado: Error de Contraseña o Sintaxis Si después de editar el archivo el robot sigue sin conectarse, puedes forzar una conexión manual para ver el error exacto en la terminal.

1. **Ejecutar comando de depuración:**

   ```
   sudo wpa_supplicant -c /etc/wpa_supplicant.conf -i wlan0
   ```

2. **Analizar el resultado:**

   - **Error `Invalid passphrase length`:** El estándar WPA exige que la contraseña tenga **mínimo 8 caracteres**. Si tu clave es más corta, el sistema la rechazará. Revisa que no te falte ningún número o letra.
   - **Error `Failed to parse network block`:** Indica que falta cerrar una llave `}` o una comilla `'`.
   - **Éxito (`CTRL-EVENT-CONNECTED`):** Si ves este mensaje, la conexión funciona. Presiona `Ctrl + C` y reinicia el robot con `sudo reboot`.

4.8.2. La cámara parpadea en el Dashboard

- **Síntoma:** La imagen de la cámara transmitida al Dashboard se ve intermitente, parpadea o se congela.
- **Causa:** Incompatibilidad del stream de video (MJPEG) con la forma en que el navegador Mozilla Firefox renderiza el contenido.
- **Solución:** Utilice un navegador web basado en Chromium, como **Google Chrome** o **Microsoft Edge**, para acceder al Dashboard. Evite el uso de Firefox para esta funcionalidad específica.

4.8.3. Las luces LED y otros componentes I2C fallan

- **Síntoma:** Los LEDs muestran colores erráticos, la cámara parpadea, o componentes como el IMU y el sensor ToF no son detectados. La terminal del robot puede mostrar errores `tegra-i2c: no acknowledge`.
- **Diagnóstico y Solución por Niveles:**
  1. **Nivel 1: Fallo de Firmware (Solución Avanzada):** A veces, el problema no es temporal y reside en el firmware del microcontrolador de la placa HUT, que es responsable de controlar los motores y los LEDs. Re-flashear el firmware puede restaurar la funcionalidad.
     - *Resolución:* Este procedimiento es necesario si el Duckiebot no reconoce el HUT (se puede verificar en el Dashboard) o si los motores no se mueven a pesar de que el sistema indica que se están enviando las señales correctamente. Para hacerlo, ejecuta el siguiente comando en tu computadora:

       ```
       dts duckiebot hut_upgrade <ROBOT_NAME>
       ```

       El proceso te guiará y te pedirá que compares la salida en la terminal con un resultado esperado para verificar que todo se ejecutó correctamente.
  2. **Nivel 2: Errores Inofensivos (Contexto de la Comunidad):** Según la comunidad de Duckietown, algunos errores `tegra-i2c` son una "peculiaridad"del kernel y pueden ser ignorados si el resto de los componentes funciona.
     - *Contexto:* A menudo, estos errores se asocian a conflictos con el bumper frontal en algunos modelos. La recomendación comunitaria es conectar el sensor ToF directamente a la placa HUT (Nota: en el modelo DB21J esto ya es así por defecto).

4.8.4. Errores comunes con `dts code workbench` (LX Braitenberg)

- **Contexto:** Al intentar probar un proyecto de código (como el LX Braitenberg) directamente en el robot usando el entorno de desarrollo, pueden surgir varios problemas de configuración.
- **Comando problemático:**

  ```
  dts code workbench -R [nombre_del_robot] -m
  ```

Error 1: Fallo de conexión SSH (Permission denied)

- **Síntoma:** El terminal muestra `Permission denied (publickey,password)` o `ssh_-askpass: exec(...): No such file or directory`.
- **Causa:** El comando `dts` no puede gestionar contraseñas interactivas. Necesita autenticación automática mediante una clave SSH. Aunque puedas conectarte manualmente con contraseña, `dts` no puede.
- **Solución:** Configurar una clave SSH para el usuario `duckie` en el robot.
  1. **Verificar si existe una clave:** `ls /.ssh/id_rsa.pub`

2. **Si no existe, crear una:** `ssh-keygen -t rsa -b 4096`
3. **Agregar la clave al Duckiebot** (pedirá la contraseña por última vez):

```
ssh-copy-id duckie@myduckiebot00.local
```

4. **Probar el acceso sin contraseña:** `ssh duckie@myduckiebot00.local` (Debe conectar directamente).

Error 2: The command mutagen is required

- **Síntoma:** El terminal muestra `The command 'mutagen' is required by 'dts devel run'`.
- **Causa:** `mutagen` es la herramienta esencial que sincroniza en tiempo real los archivos de tu proyecto entre tu PC y el Duckiebot. No está instalada en tu computadora.
- **Solución:** Instalar Mutagen en tu PC.
  1. **Instalar con el script:**

```
curl -sS https://webi.sh/mutagen | sh
```

  2. **Actualizar la variable PATH** en la sesión actual (o reiniciar el terminal):

```
source ~/.config/envman/PATH.env
```

  3. **Verificar la instalación:** `mutagen version`

Advertencia 3: pkg_resources is deprecated

- **Síntoma:** Aparece una advertencia `UserWarning: pkg_resources is deprecated as an API...`.
- **Causa:** Esto no es un error fatal. Es una advertencia de Python sobre una biblioteca que Duckietown usa internamente (`setuptools`) y que será eliminada en futuras versiones.
- **Solución (Opcional):** Si el aviso es molesto, se puede .ªnclarüna versión anterior de `setuptools` en tu entorno de Python:

```
pip install "setuptools<81"
```

Advertencia 4: Found orphan containers

- **Síntoma:** El log muestra `WARN[0000] Found orphan containers (...) for this project`.
- **Causa:** Hay contenedores de Docker de proyectos o ejecuciones antiguas que todavía están activos y no pertenecen a este nuevo despliegue.
- **Solución:** Limpiar los contenedores huérfanos.

```
docker compose down --remove-orphans
```

4.8.5. Error de Permisos `rsync` (code 23) al Sincronizar Código

- **Contexto:** Al ejecutar `dts code workbench -R [nombre_del_robot]` para sincronizar un proyecto de código local con el robot.
- **Síntoma:** El comando falla con un error de `rsync` (code 23). El log muestra errores como `Permission denied (13)` y `Operation not permitted (1)`.

```
rsync: chgrp "/code/lx-kinematics-odometry/packages" failed: Operation not permitted (1
rsync: recv_generator: mkdir "/code/..." failed: Permission denied (13)
...
rsync error: some files/attrs were not transferred (code 23)
...
subprocess.CalledProcessError: Command 'rsync ...' returned non-zero exit status 23.
```

- **Causa:** El comando `dts code workbench` utiliza `rsync` para copiar los archivos de tu proyecto a la carpeta `/code/` dentro del Duckiebot. El error indica que el usuario `duckie` (con el que `dts` se conecta por SSH) no tiene permisos de escritura en ese directorio, ya que probablemente pertenece al usuario `root`.
- **Solución:** Conectarse al Duckiebot por SSH y cambiar el propietario de la carpeta `/code/` para que pertenezca al usuario `duckie`.

  1. **Acceder al robot por SSH:**

     ```
     ssh duckie@myduckiebot01.local
     ```

     (Contraseña por defecto: `quackquack`)

  2. **Cambiar el propietario de la carpeta:**
     Usar `sudo` para cambiar el propietario (owner) de la carpeta `/code` y todo su contenido (`-R`) al usuario y grupo `duckie`.

     ```
     sudo chown -R duckie:duckie /code
     ```

  3. **Salir y reintentar:** Sal de la sesión SSH (`exit`) y vuelve a ejecutar el comando `dts code workbench` en tu computadora. Ahora `rsync` tendrá los permisos necesarios.

5. Calibración de la Cámara

La calibración de la cámara es un paso fundamental para que el Duckiebot pueda interpretar correctamente su entorno. Se divide en dos procesos clave: la calibración intrínseca (parámetros internos de la cámara) y la calibración extrínseca (pose relativa al robot y al suelo).

5.1. Introducción a la Calibración de la Cámara y Tablero

Cada cámara es única debido a las diferencias de fabricación y ensamblaje. Por lo tanto, es esencial realizar un procedimiento de calibración para compensar estas discrepancias.

Este proceso utiliza un patrón predeterminado (el tablero de calibración) para resolver los parámetros internos y externos de la cámara.

Requisitos del Tablero de Calibración (Físico) Si no cuentas con el tablero de calibración de Duckietown:

- Descargar Patrón de Calibración A3 (PDF)
- Imprímelo en formato **A3**.
- **Verifica la medida:** Los lados de los cuadrados deben tener exactamente **0.031 m (3.1 cm)**. Mide esto, ya que un tamaño incorrecto puede llevar a que tu Duckiebot choque.
- Fíjalo a una superficie **rígida y plana** (*Warning: si el patrón no es rígido, las calibraciones no deben usarse*).

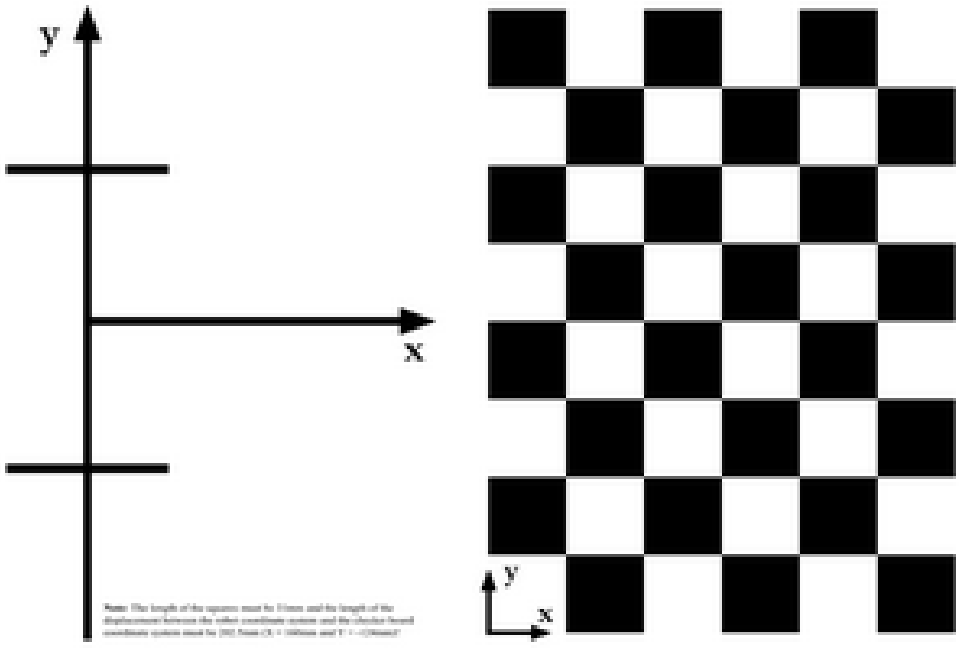

Figura 3: Patrón de calibración de la cámara de Duckietown (Fig. 43).

5.2. Calibración Intrínseca (Robot Físico)

El procedimiento de calibración intrínseca identifica parámetros que permiten crear una relación entre el plano de la imagen 2D y el mundo 3D, corrigiendo la distorsión de la lente (efecto ojo de pez).

5.2.1. El Calibrador Intrínseco (`Intrinsics Calibrator`)

Para realizar la calibración intrínseca en un robot real, ejecuta:

```
dts duckiebot calibrate_intrinsics DUCKIEBOT_NAME
```

Teclas de Funcionalidad

- **X/Z:** Aumentar/Disminuir Tasa de Fotogramas.
- **Space:** Calibrar la cámara.
- **F:** Alternar vista `Undistort`.
- **R:** Refrescar ventana.

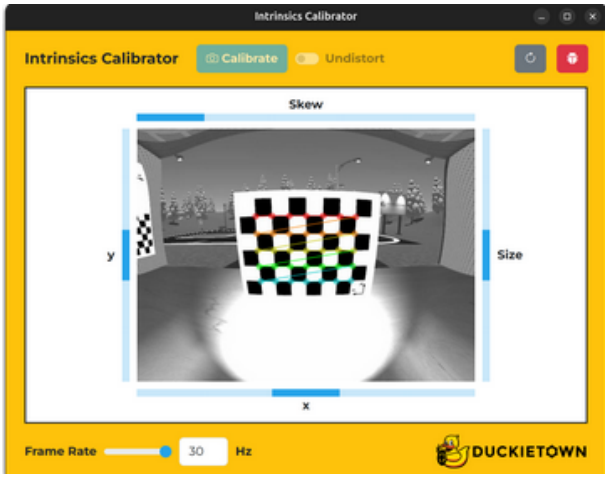

Figura 4: Interfaz del calibrador intrínseco de Duckietown (Fig. 44).

5.3. Calibración Intrínseca en Duckiematrix (Robot Virtual)

Para realizar la calibración en un Duckiebot virtual, el procedimiento requiere interactuar con el entorno de simulación 3D y seguir una secuencia específica de comandos para asegurar la compatibilidad.

1. **Abrir la simulación Matrix:** Inicia el entorno de Duckiematrix en modo *standalone* con el mapa *sandbox*:

```
dts matrix run --standalone --embedded --map sandbox
```

2. **Adjuntar el Duckiebot:** En una nueva terminal, conecta tu robot virtual a la entidad del vehículo en el mapa:

```
dts matrix attach myduckiebot00 map_0/vehicle_0
```

3. **Ingresar al Laboratorio:** Navega en la simulación con las flechas del teclado. Busca el edificio del laboratorio/garaje y presiona la tecla **F** para ingresar.

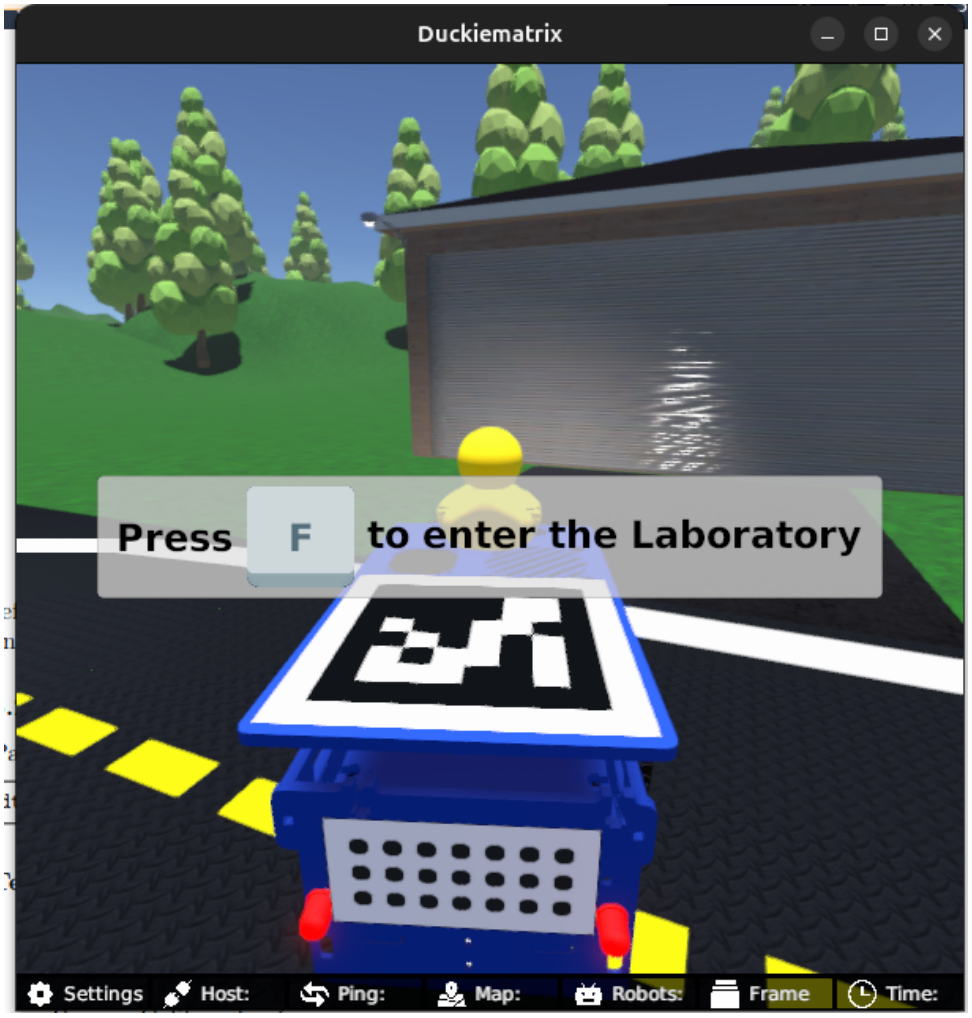

Figura 5: Entrada al laboratorio de calibración en Duckiematrix.

4. **Ejecutar el comando de calibración:** Una vez dentro del laboratorio, ejecuta el siguiente comando en tu terminal (es crítico usar la versión de API específica si estás en Ubuntu 24.04+):

```
DOCKER_API_VERSION=1.44 dts duckiebot calibrate_intrinsics myduckiebot00
```

5. **Realizar el procedimiento "El Baile"):** Dentro de la simulación, deberás mover el panel de calibración virtual usando los controles en pantalla hasta llenar completamente las barras de progreso (X, Y, Size, Skew).

6. **Verificación Visual (Undistort):** Antes de finalizar, presiona el botón **Undistort** en la herramienta de calibración.

    - Debes observar que la imagen del tablero se vuelve plana y las líneas curvas (efecto ojo de pez) desaparecen, eliminando la distorsión angular.

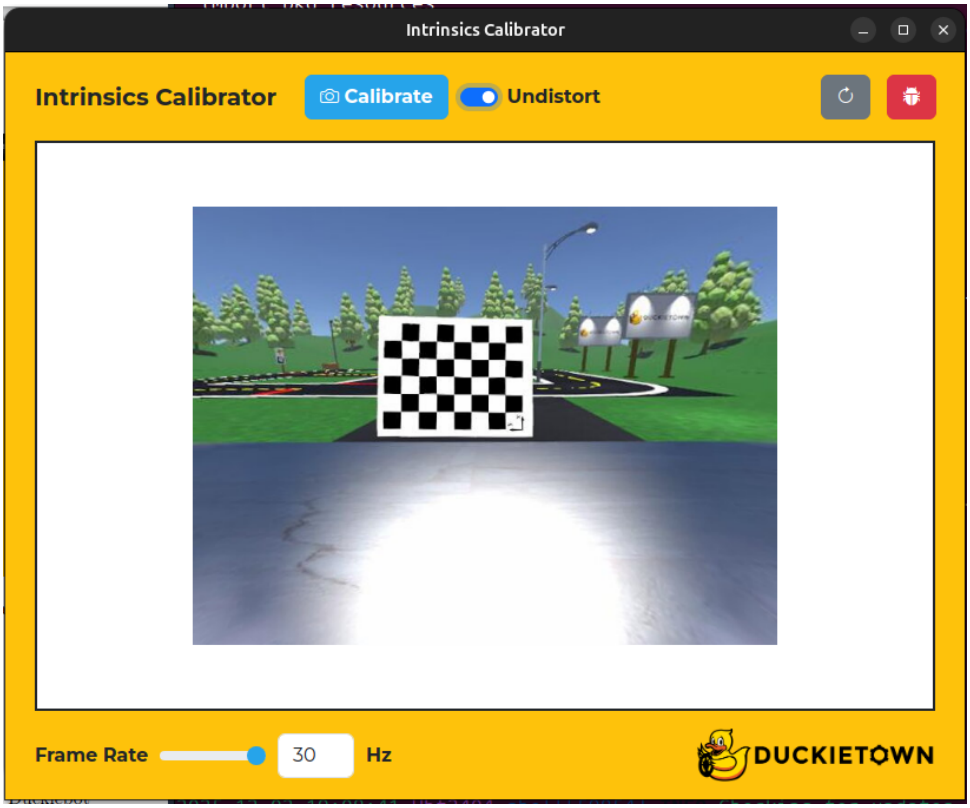

Figura 6: Vista con corrección de distorsión (Undistort) activada.

7. **Guardar Calibración:** Si la imagen se ve correcta, presiona el botón **Calibrate** y espera a que finalice el proceso.

8. **Verificación final:** Confirma que el archivo se guardó correctamente abriendo el Dashboard del robot:

```
dts duckiebot dashboard myduckiebot00 --page robot/calibrations
```

   Deberías ver una marca verde de confirmación en la sección *Camera Intrinsic*.

5.4. Calibración Extrínseca

Esta calibración genera la pose relativa entre el marco de la cámara y el marco del Duckiebot.

5.4.1. El Calibrador Extrínseco (`Extrinsics Calibrator`)

Para realizar la calibración extrínseca en un robot físico, ejecuta:

```
dts duckiebot calibrate_extrinsics DUCKIEBOT_NAME
```

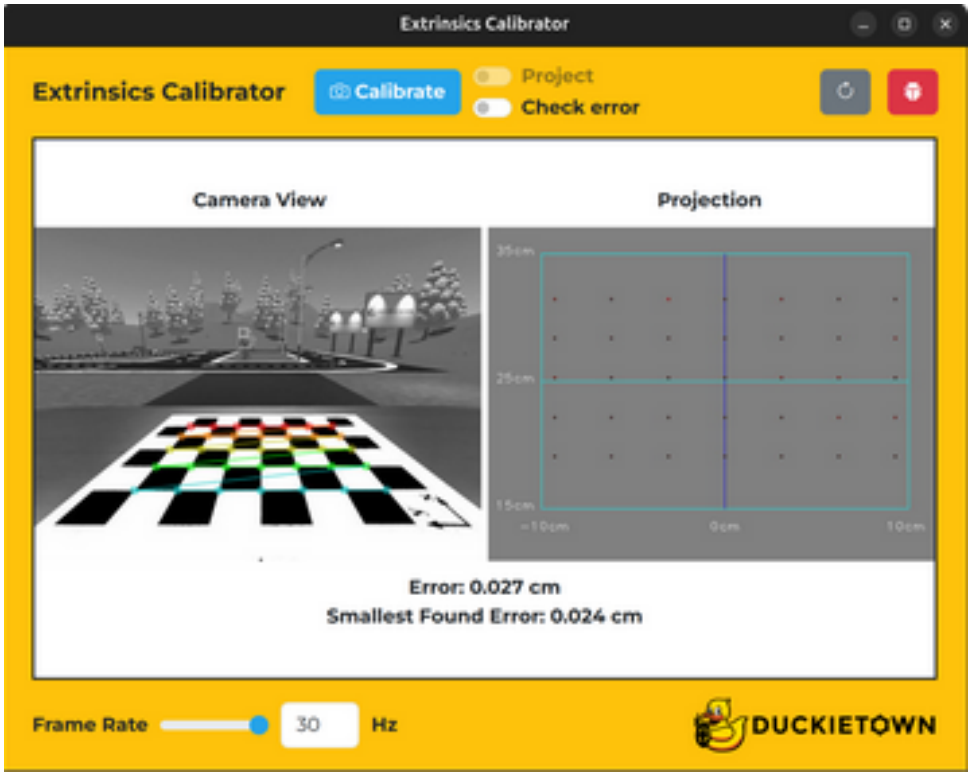

Figura 7: Interfaz del calibrador extrínseco de Duckietown (Fig. 48).

5.4.2. Procedimiento de Calibración Extrínseca (Físico)

1. (Opcional) Haz clic en **Check error** (**E**) para ver el trapezoide de alineación.
2. **Posiciona el Duckiebot** en el tablero de modo que:
   - El eje de las ruedas esté alineado con el eje Y del patrón.
   - El centro del robot esté alineado con el eje X del patrón.
3. Haz clic en **Calibrate** (**Space**).

5.4.3. Calibración Extrínseca en Duckiematrix (Virtual)

El proceso es similar a la calibración intrínseca virtual, pero seleccionando la herramienta adecuada dentro del laboratorio.

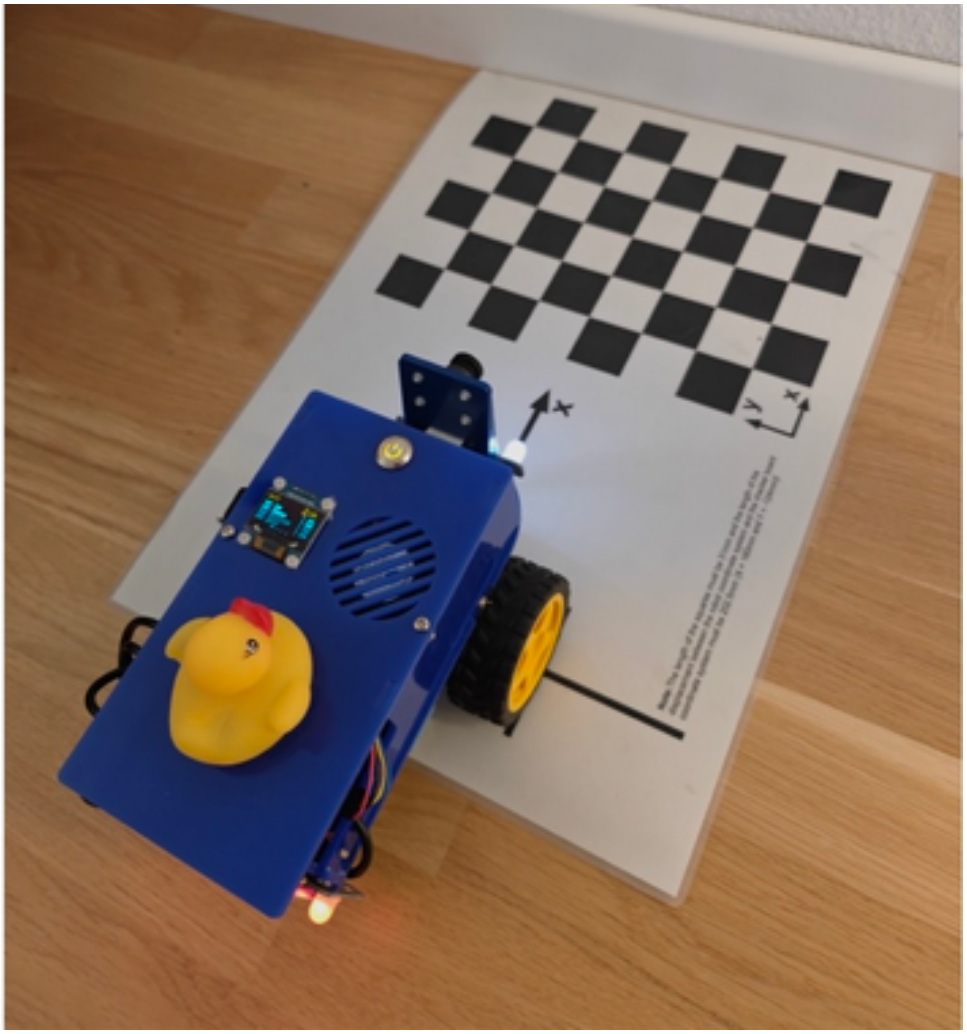

Figura 8: Configuración para la calibración extrínseca (Fig. 49).

1. Inicia la simulación (Matrix), adjunta el robot y entra al laboratorio (presionando **F** en la entrada del garaje), tal como se describió en la sección anterior.
2. Una vez dentro, selecciona la opción **Extrinsics Tool** en el menú de la simulación.
3. Ejecuta el comando de calibración apuntando a tu robot virtual:

   ```
   DOCKER_API_VERSION=1.44 dts duckiebot calibrate_extrinsics VIRTUAL_BOT_NAME
   ```

4. Usa los controles del teclado para posicionar el robot virtual sobre el patrón virtual, alineando los ejes tal como se haría en la realidad.
5. Presiona **Calibrate** en la interfaz web.

Verificación de Calibración

Para confirmar que se han creado los archivos (tanto en físico como virtual), revisa el Dashboard:

```
dts duckiebot dashboard DUCKIEBOT_NAME --page robot/calibrations
```

6. Calibración de Ruedas (Cinemática)

Para realizar el procedimiento de calibración cinemática para tu Duckiebot se utiliza el Controlador de Teclado (Keyboard Controller).

Nota Importante de Mantenimiento: Antes de iniciar la calibración de ruedas, es **crítico** verificar dos aspectos físicos del robot para evitar una mala calibración:

- **Rigidez de Motores:** Verifique que los tornillos que sujetan los motores al chasis estén correctamente apretados. Los motores no deben moverse ni vibrar independientemente del chasis. Una estructura floja invalidará la calibración.
- **Estado de la Batería:** Asegúrese de tener la batería completamente cargada. Una batería con baja carga entrega menos potencia, afectando el comportamiento de los motores y el resultado del *trim*.

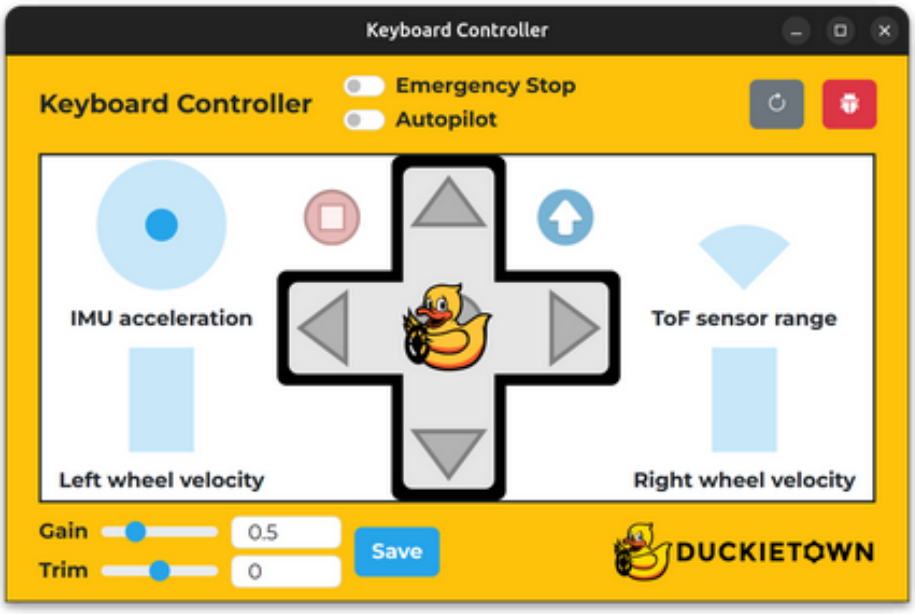

Figura 9: El controlador de teclado permite ajustar los parámetros de calibración de las ruedas.

Para iniciar la herramienta, ejecuta:

```
dts duckiebot keyboard_control DUCKIEBOT_NAME
```

6.1. Procedimiento Físico

Para realizar el procedimiento de calibración cinemática en el mundo real:

1. Crea una línea recta de algo más de 2 m de largo en el suelo usando cinta adhesiva (o usa 4 baldosas rectas como referencia).
2. Coloca tu Duckiebot en un extremo de la línea.
3. Toma nota de la posición de tu Duckiebot.
4. Orienta tu Duckiebot hacia el otro extremo de la línea.

5. Conduce tu Duckiebot hacia adelante unos 2 m.
6. Toma nota de la posición final de tu Duckiebot.
7. Mide la distancia entre el centro de la cinta y el centro del eje de tu Duckiebot usando una regla (perpendicular a la cinta).
8. **Ajuste del Trim:** Disminuye (aumenta) el valor de *Trim* usando las teclas indicadas en la terminal y repite los pasos 2-7 si tu Duckiebot se desvió hacia la izquierda (derecha) de la cinta por más de 10 cm.
9. (Opcional) Establece la Ganancia (Gain) si la velocidad general es muy alta o baja.
10. Presiona la barra espaciadora (**Guardar**) para escribir los cambios en el robot.

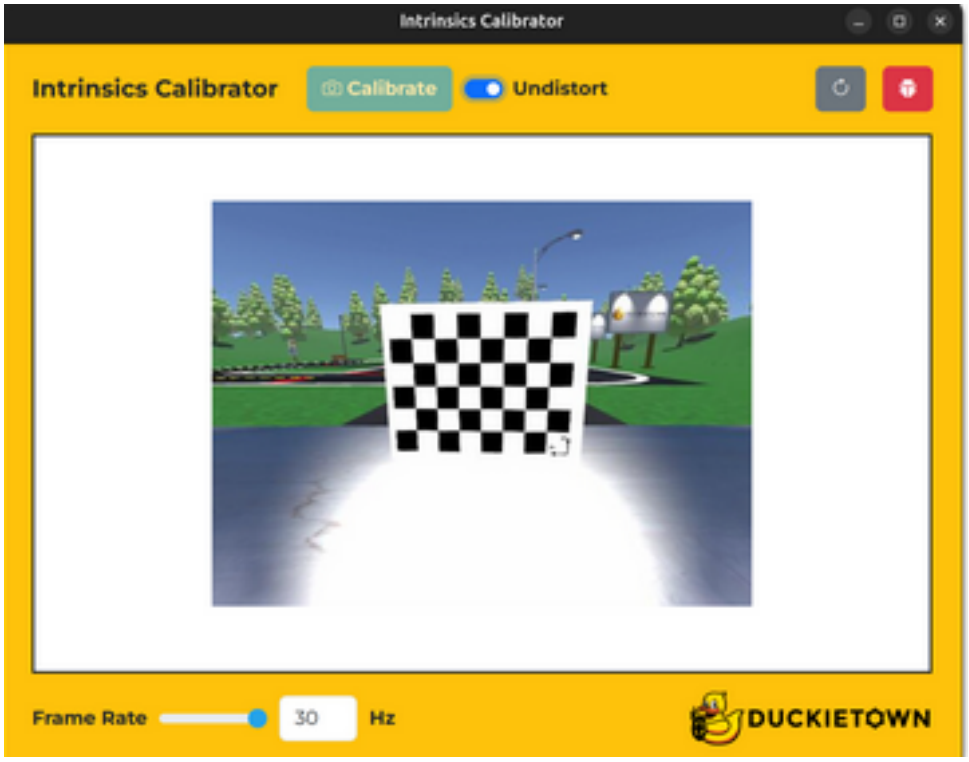

Figura 10: Marca una línea recta en el suelo para establecer una línea base.

6.2. Procedimiento Virtual (Duckiematrix)

Es posible calibrar la cinemática de un robot virtual simulando el mismo proceso.

1. Inicia la Duckiematrix en el mapa **sandbox**, el cual tiene líneas rectas y cuadrículas en el suelo:

```
dts matrix run --standalone --embedded --map sandbox
```

2. Inicia el controlador de teclado apuntando a tu robot virtual:

```
dts duckiebot keyboard_control VIRTUAL_BOT_NAME
```

3. Conduce el robot virtual a lo largo de una de las líneas del carril en el mapa *sandbox*.

4. Observa visualmente la desviación respecto a la línea del suelo.
5. Ajusta el *Trim* en la terminal de la misma manera que en el robot físico hasta que el robot virtual conduzca recto.
6. Guarda la configuración presionando la barra espaciadora.

6.3. Verificación Final

Para confirmar que se ha creado un nuevo archivo de calibración cinemática, ejecuta el siguiente comando e inspecciona el contenido del panel Kinematics:

```
dts duckiebot dashboard DUCKIEBOT_NAME --page robot/calibrations
```

Nota: Dentro del panel Kinematics, en Local, deberías ver una marca (tick) junto a Completed, la fecha de calibración y la ruta /data/config/calibrations/kinematics/DUCKIEBOT_NAME.yaml.

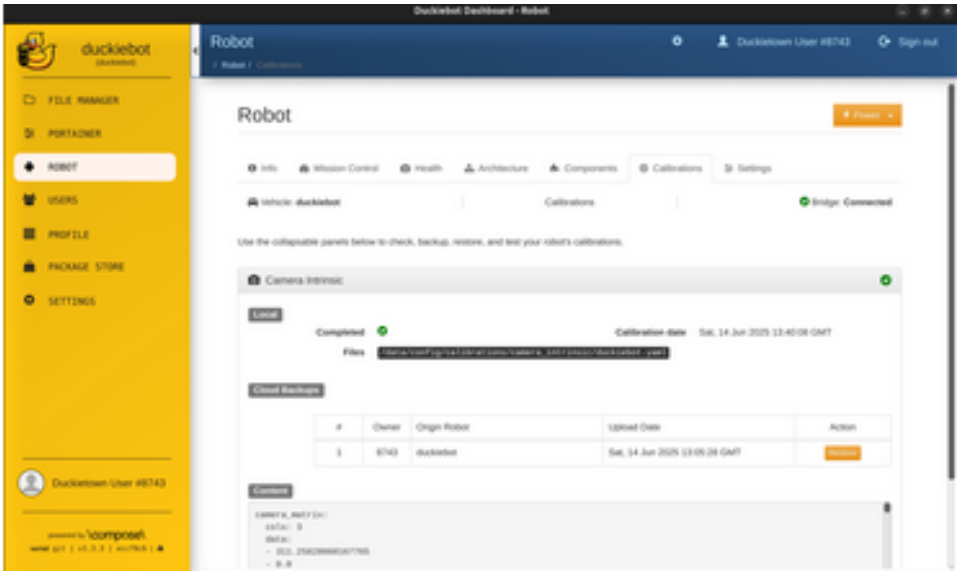

Figura 11: La deriva (drift) es causada por ruedas no calibradas.

7. Instalación y Ejecución de la Duckiematrix

7.1. Requisitos Previos

- Una instalación funcional de Duckietown Shell: Duckietown Shell (dts).

7.2. Qué obtendrás

- Una instalación exitosa de la Duckiematrix.
- Conocimiento sobre cómo iniciar la Duckiematrix.

7.3. Cómo instalar la Duckiematrix

La Duckiematrix se instala con Duckietown Shell usando el comando:

```
dts matrix install
```

7.4. Cómo Ejecutar la Duckiematrix

El comando básico para ejecutar la Duckiematrix es `dts matrix run`. Sin embargo, para ejecutar la Duckiematrix necesitas proporcionar dos piezas de información:

- Dónde se está ejecutando el motor (*engine*).
- Qué mapa cargar.

En la mayoría de los casos, probablemente querrás ejecutar el motor localmente. Puedes hacerlo añadiendo la bandera `-standalone`.

Respecto al mapa, puedes elegir entre definir tu propio mapa e indicarle a la Duckiematrix dónde encontrarlo en el sistema de archivos, o puedes usar uno de los mapas predeterminados que se proporcionan incluyendo la bandera `-embedded`.

Los mapas predeterminados se llaman:

- `sandbox`
- `demo`
- `empty`
- `intersections`
- `loop`

En resumen, el siguiente comando:

```
dts matrix run --standalone --embedded --map sandbox
```

...iniciaría la Duckiematrix en tu máquina local y cargaría el mapa `sandbox` predefinido.

Por defecto, la Duckiematrix se ejecuta en modo tutorial, donde se introducen las combinaciones de teclas. Si quieres deshabilitar este comportamiento, puedes añadir la bandera `-no-tutorial`. Si algo va mal y no ves que el renderizador se inicializa, puedes ver qué está pasando añadiendo la bandera `-verbose`.

7.5. Usando un Motor Remoto

Para ejecutar un renderizador local y conectarlo a un motor remoto, simplemente necesitas especificar la ubicación del motor (como un nombre de host o dirección IP) usando la bandera `-engine`:

```
dts matrix run --engine ENGINE_HOSTNAME
```

En este caso no necesitas especificar un mapa, ya que eso se especificó cuando el motor de la Duckiematrix se ejecutó inicialmente.

7.6. Conectando un Duckiebot a un Motor Remoto

Una vez que tengas un motor de Duckiematrix (Engine) en funcionamiento, puedes conectar un Duckiebot (físico o virtual) a una entidad en la Duckiematrix. Puedes hacerlo con el comando:

```
dts matrix attach [--engine ENGINE_HOSTNAME] DUCKIEBOT_NAME ENTITY_NAME
```

...donde puedes omitir `ENGINE_HOSTNAME` si el motor se está ejecutando localmente. Puedes encontrar el nombre de la entidad de la matrix (*matrix entity*) mirando en las configuraciones del mapa o, más simple, haciendo clic en la pestaña *Robots* en la parte inferior de la ventana de renderizado de Duckiematrix y luego buscando el *Nombre (Name)*. El valor predeterminado para un Duckiebot es `map_0/vehicle_0`, por lo que suele ser una buena suposición.

7.7. Usando un Mapa Personalizado

El mapa se puede definir en el sistema de archivos local y ser cargado por el motor. Para aprender cómo definir un mapa local, consulta la página sobre creación de mapas. Este mapa puede ser cargado omitiendo la bandera `-embedded` y especificando la ruta:

```
dts matrix run --standalone --map PATH_TO_MAP
```

8. Duckiebots Virtuales

8.1. Qué necesitarás

- Una instalación exitosa de la Duckiematrix: Cómo instalar la Duckiematrix.

8.2. Qué obtendrás

- Conocimiento sobre los robots virtuales de Duckietown.

8.3. Introducción

Los robots virtuales de Duckietown (Virtual Duckietown robots) permiten que la pila completa de software de un robot Duckietown se ejecute en una máquina local en su propio entorno Docker, permitiendo la simulación completa de cualquier aspecto de ese robot Duckietown, lo que habilita pruebas de integración.

Una vez que un robot virtual Duckietown está en funcionamiento, se comportará de acuerdo con su equivalente físico, incluyendo cómo responde a los comandos `dts`.

Lo más común que probablemente querrás hacer es crear un robot virtual y luego iniciarlo. Una vez que esté en marcha (lo verás cuando ejecutes el comando `dts fleet discover` como cualquier otro Duckiebot), probablemente querrás adjuntarlo a una entidad en la Duckiematrix. Una vez que esta conexión esté completa, todos los datos recolectados por la entidad en la Duckiematrix serán enviados al robot virtual y, a la inversa, cualquier acción de control que tomes con tu Duckiebot virtual será ejecutada en la entidad de la Duckiematrix.

8.4. La API de `dts` para Robots Virtuales

Como la mayoría de las cosas en Duckietown, la forma principal de realizar operaciones en robots virtuales es a través de Duckietown Shell (`dts`). A continuación, se muestra una lista de operaciones que puedes realizar:

Crear (Create)

Para crear el robot virtual Duckietown `ROBOT_NAME`, ejecuta el siguiente comando, donde `TYPE` y `CONFIGURATION` son su tipo y configuración, respectivamente:

```
dts duckiebot virtual create --type TYPE --configuration CONFIGURATION ROBOT_NAME
```

por ejemplo:

```
dts duckiebot virtual create --type duckiebot --configuration DB21J myduckiebot00
```

Iniciar (Start)

Para iniciar el robot virtual Duckietown `ROBOT_NAME`, ejecuta:

```
dts duckiebot virtual start ROBOT_NAME
```

Conectar (Connect)

Para conectarse al robot virtual Duckietown `ROBOT_NAME`, ejecuta:

```
dts duckiebot virtual connect ROBOT_NAME
```

Listar (List)

Para listar los robots virtuales Duckietown existentes y sus estados, ejecuta:

```
dts duckiebot virtual list
```

Reiniciar (Restart)

Para reiniciar el robot virtual Duckietown `ROBOT_NAME`, ejecuta:

```
dts duckiebot virtual restart ROBOT_NAME
```

Detener (Stop)

Para detener el robot virtual Duckietown `ROBOT_NAME`, ejecuta:

```
dts duckiebot virtual stop ROBOT_NAME
```

Destruir (Destroy)

Para destruir el robot virtual Duckietown `ROBOT_NAME` y eliminar todas sus imágenes de Docker, ejecuta:

```
dts duckiebot virtual destroy ROBOT_NAME
```

8.5. Adjuntando (Attaching) Robots Virtuales

Para que un robot Duckietown actúe y perciba dentro de la Duckiematrix, necesita un proxy dentro de la Duckiematrix (una entidad de la Duckiematrix) al cual adjuntarse. Se dice que un robot Duckietown fuera de la Duckiematrix está *adjunto* (attached) a una entidad de la Duckiematrix cuando todos sus sensores y actuadores están vinculados a sus contrapartes virtuales dentro de la Duckiematrix.

Para adjuntar el robot Duckietown `ROBOT_NAME` a la entidad de la Duckiematrix `ENTITY_NAME` (p.ej., `map_0/vehicle_0`), ejecuta el siguiente comando, donde `ENGINE_HOSTNAME` es el nombre de host (o dirección IP) opcional del Motor (Engine):

```
dts matrix attach [--engine ENGINE_HOSTNAME] ROBOT_NAME ENTITY_NAME
```

por ejemplo:

```
dts matrix attach myduckiebot00 map_0/vehicle_0
```

Nota Esto aplica tanto para robots Duckietown físicos como virtuales.

Consejo Usa la opción `-dreamwalk` para habilitar el *dreamwalking* en Duckiebots físicos (es decir, los comandos se enviarán tanto a sus actuadores físicos como a sus contrapartes en la Duckiematrix).

Atención Asegúrate de que el firewall de tu computadora esté inactivo o que se haya añadido una regla de permiso para `ENGINE_HOSTNAME`. Para mostrar el estado del firewall de tu computadora, ejecuta:

`sudo ufw status`

Para añadir una regla de permiso para `ENGINE_HOSTNAME` al firewall de tu computadora, ejecuta:

`sudo ufw allow ENGINE_HOSTNAME`

8.6. Controladores (Drivers) Soportados

Los controladores (drivers) de robots virtuales Duckietown permiten la comunicación entre la pila ROS de un robot Duckietown y una entidad dentro de la Duckiematrix.

Estado de Implementación

Controlador (Driver)	Implementado
Cámara	Sí
Tiempo de Vuelo (Time-of-Flight)	Sí
IMU	Sí
LED	Sí
Encoder	Sí
Rueda (Wheel)	Sí

Figura 12: Tipos de datos intercambiados con la Duckiematrix por los controladores del robot virtual Duckietown.

9. Creación de Mapas Personalizados: El Circuito del Laboratorio

Esta sección detalla la configuración exacta para recrear digitalmente el circuito físico disponible en el laboratorio dentro de Duckiematrix .

9.1. Recursos y Estructura de Archivos

Para facilitar la configuración, puedes descargar un ejemplo base de un mapa de circuito cerrado (loop) desde el siguiente enlace:

Descargar Mapa de Ejemplo (Loop.zip)

Un mapa en Duckiematrix se compone de un directorio que contiene archivos `.yaml` . Para definir nuestro circuito, necesitamos dos archivos principales:

- `tiles.yaml`: Define qué tipo de baldosa va en cada posición (curva, recta, intersección) .
- `frames.yaml`: Define las coordenadas exactas y la orientación de cada baldosa y del robot .

9.2. Definición de Tipos (`tiles.yaml`)

El siguiente código corresponde a la distribución de 5x3 baldosas del laboratorio . Guarde este contenido en un archivo llamado `tiles.yaml`.

```
version: 1.0
tiles:
  # --- Fila 0 (Inferior) ---
  map_0/tile_0_0:
    type: curve
  map_0/tile_1_0:
    type: straight
  map_0/tile_2_0:
    type: 3way
  map_0/tile_3_0:
    type: straight
  map_0/tile_4_0:
    type: curve

  # --- Fila 1 (Central) ---
```

```
  map_0/tile_0_1:
    type: straight
  map_0/tile_1_1:
    type: floor
  map_0/tile_2_1:
    type: straight
  map_0/tile_3_1:
    type: floor
  map_0/tile_4_1:
    type: straight

  # --- Fila 2 (Superior) ---
  map_0/tile_0_2:
    type: curve
  map_0/tile_1_2:
    type: straight
  map_0/tile_2_2:
    type: 3way
  map_0/tile_3_2:
    type: straight
  map_0/tile_4_2:
    type: curve
```

9.3. Definición de Posiciones (`frames.yaml`)

Este archivo coloca las baldosas en su lugar y define la posición inicial del vehículo (`vehicle_0`). Observe que el robot está configurado para aparecer en coordenadas específicas ($x = 0{,}88, y = 0{,}185$). Guarde esto como `frames.yaml`.

```
version: 1.0
frames:
  # --- Definición del Mapa Base ---
  map_0:
    relative_to: ~
    pose: {x: 0.0, y: 0.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 0.0}

  # --- Posición Inicial del Robot ---
  map_0/vehicle_0:
    relative_to: ~
    pose:
      x: 0.88
      y: 0.185
      z: 0.0
      roll: 0.0
      pitch: 0.0
      yaw: 0.0

  # --- Fila 0 (Inferior) ---
  map_0/tile_0_0:
    relative_to: ~
    unit: tiles
    pose: {x: 0.0, y: 0.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: -1.5707963}
  map_0/tile_1_0:
    relative_to: ~
    unit: tiles
    pose: {x: 1.0, y: 0.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 1.5707963}
  map_0/tile_2_0:
    relative_to: ~
    unit: tiles
    pose: {x: 2.0, y: 0.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 0.0}
  map_0/tile_3_0:
    relative_to: ~
    unit: tiles
    pose: {x: 3.0, y: 0.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 1.5707963}
  map_0/tile_4_0:
    relative_to: ~
    unit: tiles
    pose: {x: 4.0, y: 0.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 0.0} # 90 deg

  # --- Fila 1 (Central) ---
  map_0/tile_0_1:
    relative_to: ~
    unit: tiles
    pose: {x: 0.0, y: 1.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 0.0} # 270 deg
  map_0/tile_1_1:
    relative_to: ~
```

```
    unit: tiles
    pose: {x: 1.0, y: 1.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 0.0}
  map_0/tile_2_1:
    relative_to: ~
    unit: tiles
    pose: {x: 2.0, y: 1.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 0.0}
  map_0/tile_3_1:
    relative_to: ~
    unit: tiles
    pose: {x: 3.0, y: 1.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 0.0}
  map_0/tile_4_1:
    relative_to: ~
    unit: tiles
    pose: {x: 4.0, y: 1.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 0.0} # 90 deg

  # --- Fila 2 (Superior) ---
  map_0/tile_0_2:
    relative_to: ~
    unit: tiles
    pose: {x: 0.0, y: 2.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 3.1415926}
  map_0/tile_1_2:
    relative_to: ~
    unit: tiles
    pose: {x: 1.0, y: 2.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 1.5707963}
  map_0/tile_2_2:
    relative_to: ~
    unit: tiles
    pose: {x: 2.0, y: 2.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 3.1415926}
  map_0/tile_3_2:
    relative_to: ~
    unit: tiles
    pose: {x: 3.0, y: 2.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 1.5707963}
  map_0/tile_4_2:
    relative_to: ~
    unit: tiles
    pose: {x: 4.0, y: 2.0, z: 0.0, roll: 0.0, pitch: 0.0, yaw: 1.5707963}
```

9.4. Ejecución del Mapa

Para cargar este mapa específico en Duckiematrix:

1. Cree una carpeta llamada `mapa_laboratorio`.
2. Guarde los dos códigos anteriores dentro de esa carpeta como `tiles.yaml` y `frames.yaml`.
3. Ejecute el siguiente comando apuntando a la ruta de la carpeta:

```
dts matrix run --standalone --map /ruta/a/mapa_laboratorio
```

10. Procedimientos Generales para Experiencias de Aprendizaje (LX)

La mayoría de las Experiencias de Aprendizaje (LX) en Duckietown siguen un flujo de trabajo estandarizado. Esta sección detalla los pasos comunes para configurar el repositorio, mantener el sistema actualizado y ejecutar el código.

10.1. Configuración del Repositorio (Forking y Cloning)

Para trabajar en cualquier ejercicio, necesitas tu propia copia del código en GitHub.

10.1.1. 1. Crear un Fork

Navega al repositorio de la LX correspondiente en GitHub (se indicará en la sección específica de cada ejercicio). Busca y presiona el botón **'Fork'** en la parte superior derecha. Esto creará un nuevo repositorio en: `<tu_usuario_github>/lx-nombre-del-ejercicio`.

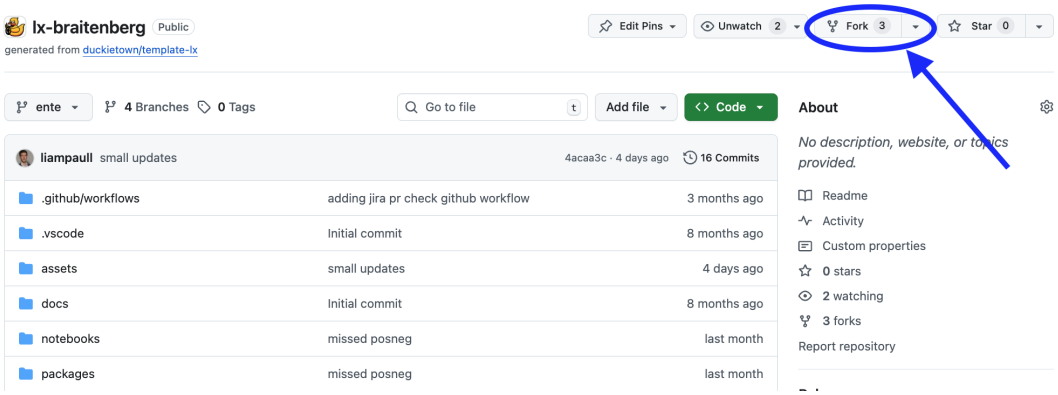

Figura 13: Hacer Fork del LX permite realizar cambios locales y recibir actualizaciones.

10.1.2. 2. Clonar el Fork

Clona el fork en tu computadora, reemplazando tu nombre de usuario y el nombre del repositorio en el comando a continuación:

```
git clone git@github.com:<tu_usuario_github>/lx-[nombre-del-ejercicio]
cd lx-[nombre-del-ejercicio]
```

10.1.3. 3. Configurar el repositorio Upstream

Debes configurar el repositorio oficial de Duckietown como **'upstream'** para sincronizar tu fork con las actualizaciones oficiales.

1. Lista el repositorio remoto actual:

   ```
   git remote -v
   ```

2. Especifica el nuevo repositorio remoto (cambia la URL según el ejercicio):

```
git remote add upstream https://github.com/duckietown/lx-[nombre-ejercicio]
```

3. Confirma que se añadió correctamente:

```
git remote -v
```

10.2. Manteniendo el Sistema Actualizado

Nota Importante: Asegúrate de que tu Duckietown Shell esté configurado en un perfil `ente`. Verifícalo con `dts profile list`.

Ejecuta los siguientes pasos antes de empezar:

- Sincroniza tu fork con el repositorio oficial:

```
git pull upstream ente
```

- Actualiza Duckietown Shell y el entorno de escritorio:

```
pipx upgrade duckietown-shell
dts update
dts desktop update
```

- Actualiza tu Duckiebot (real o virtual):

```
dts duckiebot update ROBOTNAME
```

10.3. Ejecución y Compilación

10.3.1. Lanzar el Editor de Código

Dentro del directorio raíz de la LX, ejecuta:

```
dts code editor
```

Abre la URL que aparece en la terminal. Usa la barra lateral para navegar a la carpeta `notebooks` y sigue las instrucciones secuenciales.

10.3.2. Construcción del Código (Build)

Una vez modificado el código, compílalo con:

```
dts code build -R ROBOT_NAME
```

10.4. Pruebas y Visualización

10.4.1. 1. Iniciar la Simulación (Si usas Robot Virtual)

Si no usas un robot físico, crea e inicia uno virtual y luego lanza la Matrix:

```
# 1. Crear (si no existe)
dts duckiebot virtual create --type duckiebot --configuration DB21J [VBOT]
# 2. Iniciar robot
dts duckiebot virtual start [VBOT]
# 3. Iniciar entorno Matrix
dts code start_matrix
```

10.4.2. 2. Ejecutar el Workbench (Lógica del Robot)

Para correr tu código:

- **En Duckiebot Físico:**

  ```
  dts code workbench -R [ROBOT_NAME]
  ```

- **En Duckiebot Virtual:**

  ```
  dts code workbench -m -R [VIRTUAL_ROBOT_NAME]
  ```

10.4.3. 3. Visualización (VNC)

Para ver lo que el robot 've' y acceder a herramientas gráficas:

```
dts code vnc -R [ROBOT_NAME]
```

11. LX: Computer Vision (Visión por Computadora)

Esta experiencia se centra en los fundamentos de la percepción visual robótica, desde el modelo matemático de la cámara hasta el control visual básico.

11.1. Requisitos Específicos

- **Repositorio:** `lx-computer-vision`
- **Hardware:** Duckiebot físico (opcional) o Duckiematrix (recomendado).

11.2. Resultados de Aprendizaje

Al finalizar esta experiencia, serás capaz de:

- Entender la relación matemática entre el mundo 3D y el plano de imagen 2D (coordenadas homogéneas).
- Formalizar el Modelo de Cámara Pinhole e identificar matrices intrínsecas y extrínsecas.
- Realizar calibraciones de cámara en robots físicos y virtuales.
- Aplicar homografías para la proyección de imágenes (vista de pájaro).
- Implementar filtros de imagen (Gaussian, Sobel, Canny) para reducir ruido y detectar bordes.
- Diseñar un controlador de *Visual Servoing* para mantener al robot en el carril.

11.3. Instrucciones de Ejecución

Para realizar esta LX, sigue los pasos detallados en la **Sección 10: Procedimientos Generales**.

Nota específica para esta LX: En el paso de clonación, utiliza la URL:

`https://github.com/duckietown/lx-computer-vision`

12. LX: Object Detection (Detección de Objetos)

Esta experiencia guía a través del proceso de recolección de datos, anotación automática y entrenamiento de una red neuronal para detectar objetos (patitos) usando la cámara del robot.

12.1. Requisitos Específicos

Además de la configuración estándar, necesitarás:

- **Una cuenta de Google:** Se utilizará Google Colab y Drive para el entrenamiento.
- **Una cuenta de Hugging Face:** Necesaria para acceder a modelos de ML.
- **Permiso para el modelo SAM3:** Debes llenar el formulario de solicitud en Hugging Face para usar el modelo de auto-etiquetado.
- **Repositorio:** `lx-object-detection`

Atención: La aprobación del modelo SAM3 puede tomar unos minutos, se recomienda solicitarla antes de iniciar.

12.2. Resultados de Aprendizaje

Al finalizar esta experiencia, serás capaz de:

- Utilizar PyTorch para construir redes neuronales.
- Recolectar datos de entrenamiento y crear un dataset.
- Anotar datos automáticamente utilizando modelos modernos.
- Crear, optimizar y entrenar tu propio detector basado en YOLO (v11).
- Integrar el modelo entrenado en un controlador para evitar atropellar patitos (seguridad peatonal).

12.3. Instrucciones de Ejecución

Sigue los pasos de la **Sección 10**, teniendo en cuenta las siguientes diferencias específicas para esta LX.

12.3.1. Configuración del Repositorio

Utiliza la siguiente URL para el upstream: `https://github.com/duckietown/lx-object-detection`

12.3.2. Nota sobre Pruebas en Robot Físico

Al momento de probar tu modelo entrenado en un Duckiebot físico, hay una diferencia crítica en el comando de *workbench*. Debes usar la bandera `-local`:

```
dts code workbench -R ROBOT_NAME --local
```

Nota: Esto hace que el código se ejecute en tu laptop (que tiene más potencia para procesar la red neuronal) mientras se comunica con el robot, en lugar de ejecutarse directamente en la CPU limitada del Duckiebot.

12.4. Solución de Problemas Específicos

- **Síntoma:** Error `No valid DTProject found` al abrir el editor.

  Solución: Asegúrate de ejecutar los comandos desde dentro de la carpeta raíz `lx-object-detection`.

- **Síntoma:** El robot virtual se cuelga al actualizar.

  Solución: Reinicia el contenedor con `dts duckiebot virtual restart [NOMBRE]`.

- **Síntoma:** Crash con `FileNotFoundError: ... best.onnx` al iniciar la recolección de datos.

  Causa: Error de lógica en la plantilla base; intenta cargar el modelo antes de que este exista (antes del entrenamiento).

  Solución: Modificar el archivo `model.py` agregando una verificación (`try/except` o `if os.path.exists`) para que el agente pueda iniciar sin cargar el modelo obligatoriamente.

- **Síntoma:** El agente no genera imágenes o se queda en espera indefinida.

  Causa: Dependencia del entorno visual no satisfecha. El robot no tiene "ojos" si la Matrix no está corriendo primero.

  Solución: Respetar el orden estricto de ejecución:

  1. `dts code start_matrix` (Entorno visual).
  2. `dts code workbench ... -L data_collection` (Lógica).
  3. `dts duckiebot keyboard_control ...` (Control).

- **Síntoma:** No encuentro dónde ingresar el `HF_TOKEN` en Google Colab.

  Causa: La opción de variables de entorno ("Secrets") se encuentra en un menú lateral colapsado.

  Solución: Busca el icono de la **Llave** en la barra lateral izquierda. Al añadir el secreto, asegúrate de activar el interruptor **Notebook access** (ver Figura 14).

13. LX: Localization - EKF (Localización y Filtro de Kalman)

Esta experiencia de aprendizaje aborda cómo debemos utilizar los datos que fluyen a través de los sensores, junto con el conocimiento de nuestro entorno, para estimar nuestro estado (posición y orientación).

Exploraremos aproximaciones al filtro de Bayes, como el filtro de Kalman, el filtro de partículas y el filtro de histograma. Finalmente, programarás un **Filtro de Kalman**

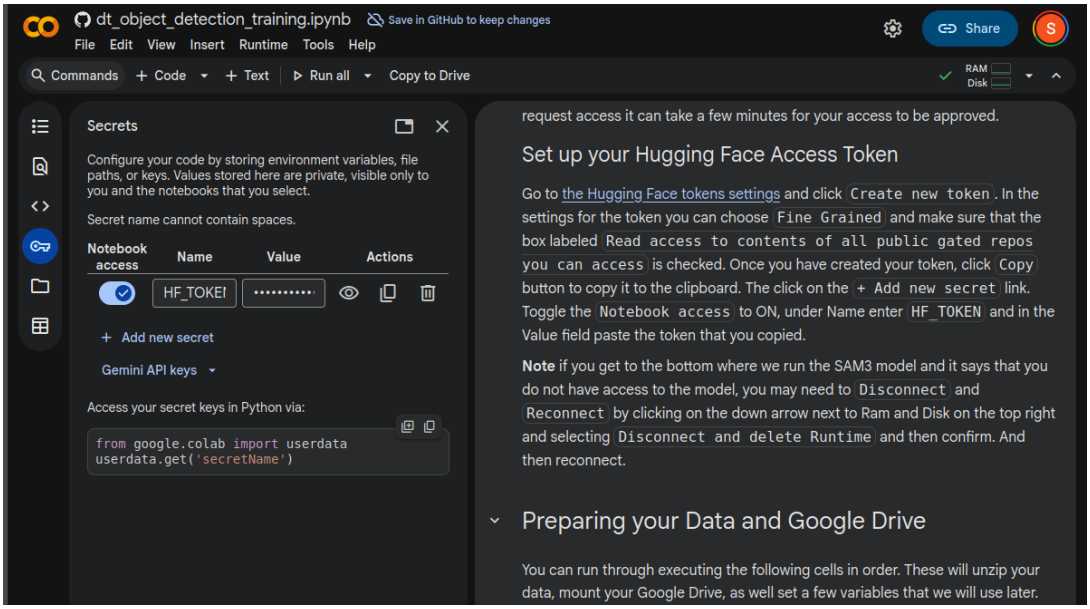

Figura 14: Ubicación del menú Secrets (Llave) y activación de acceso al Notebook.

Extendido (EKF) para localizar tu Duckiebot utilizando datos de los encoders de las ruedas y marcadores fiduciarios (AprilTags) ubicados en posiciones conocidas.

13.1. Resultados de Aprendizaje

Al finalizar esta experiencia, serás capaz de:

- Comprender la teoría e implementación de los Filtros de Kalman.
- Comprender la teoría e implementación de los Filtros de Partículas y de Histograma.
- Evaluar las ventajas y desventajas de cada enfoque según el escenario.
- **Diseñar, ajustar y probar un EKF** para lograr una localización basada en mapas, fusionando datos de odometría (encoders) y mediciones de pose visual (AprilTags).

13.2. Requisitos y Configuración

Esta LX puede ejecutarse tanto en un Duckiebot virtual dentro de Duckiematrix como en un robot físico. Sigue los procedimientos generales de la **Sección 10**, pero ten en cuenta los siguientes detalles específicos.

13.2.1. 1. Fork y Clonado del Repositorio

1. **Hacer Fork:** Ve al repositorio oficial y crea tu fork:
   `https://github.com/duckietown/lx-ekf-localization`
2. **Clonar:** Clona tu fork en la computadora de desarrollo:

```
git clone git@github.com:<TU_USUARIO>/lx-ekf-localization
cd lx-ekf-localization
```

3. **Configurar Upstream:**

```
git remote add upstream https://github.com/duckietown/lx-ekf-localization
```

13.2.2. 2. Actualización del Sistema

Asegúrate de ejecutar los comandos de actualización dentro de la carpeta del proyecto para evitar problemas de compatibilidad:

```
pipx upgrade duckietown-shell
dts update
dts desktop update
# Si usas un robot (virtual o real), actualízalo también:
dts duckiebot update NOMBRE_ROBOT
```

13.3. Ejecución del Código

13.3.1. Editor y Notebooks

Para comenzar con la teoría y la implementación guiada, inicia el editor de código desde la raíz del proyecto:

```
dts code editor
```

Abre la URL generada en tu navegador y navega a la carpeta `notebooks`. Sigue las instrucciones secuencialmente.

13.3.2. Compilación

Una vez que hayas modificado el código en los notebooks o en los archivos Python, compílalo antes de probar:

```
dts code build -R NOMBRE_ROBOT
```

13.4. Pruebas en Duckiematrix (Simulación)

Para probar la localización, necesitamos un entorno controlado con marcadores AprilTag.

13.4.1. 1. Configurar el Robot Virtual

Si aún no tienes un robot virtual, créalo e inícialo:

```
dts duckiebot virtual create --type duckiebot --configuration DB21J [VBOT]
dts duckiebot virtual start [VBOT]
```

(Reemplaza `[VBOT]` por el nombre de tu robot virtual).

13.4.2. 2. Iniciar Duckiematrix

Desde el directorio de la LX, inicia la simulación. En este ejercicio, el mapa es más complejo e incluye señales de tráfico con AprilTags.

```
dts code start_matrix
```

Nota: Si usas la versión web y los colores se ven desaturados, intenta cambiar de navegador.

Controles en la Matrix:

- **WASD:** Mueve el patito (avatar) por el mapa.
- **E:** Abordar el Duckiebot (cuando estás cerca). Una vez a bordo, WASD controla el robot.
- **R:** Reiniciar la posición del robot. **Importante:** Debes reiniciar la posición antes de cada prueba, ya que la estimación inicial del estado coincide con la posición de reset.

13.5. Ejecución del Workbench y Visualización

13.5.1. 1. Ejecutar el Agente (Workbench)

Para correr tu algoritmo de localización:

- **En Duckiematrix (Virtual):**

  ```
  dts code workbench -m -R [VBOT]
  ```

- **En Robot Físico:**

  ```
  dts code workbench -R [NOMBRE_ROBOT]
  ```

13.5.2. 2. Visualización con RViz (VNC)

Es crucial ver qué está "pensando.[el] robot. Abre el visualizador VNC:

```
dts code vnc -R [NOMBRE_ROBOT]
```

Dentro de la interfaz VNC, abre **RViz**. Deberías observar:

- **Flecha Roja:** La pose real del robot (Ground Truth).
- **Flecha Azul:** Tu estimación de pose calculada por el EKF.
- **Elipse Púrpura:** La covarianza (incertidumbre) de tu estimación. Si tu implementación es correcta, la flecha roja debería mantenerse dentro de la elipse.
- **AprilTags:** Los marcadores detectados cambiarán de color (de verde a azul) al ser procesados por la cámara.

13.6. Solución de Problemas (Troubleshooting)

- **Síntoma:** Al ejecutar `dts code editor` aparece el error:
  `dts : No valid DTProject found at ...`

Solución: Asegúrate de estar ejecutando el comando desde dentro de la carpeta raíz `lx-ekf-localization`.

- **Síntoma:** El robot virtual se cuelga indefinidamente al intentar actualizarse o iniciarse.

  Solución: Reinicia el contenedor del robot virtual con:

```
dts duckiebot virtual restart [VBOT]
```

- **Síntoma:** Los colores en la simulación web se ven extraños.

  Solución: Prueba usar un navegador diferente o verifica que la aceleración por hardware esté activada.

13.6.1. Configuración Crítica de RViz

Es frecuente que al abrir RViz la pantalla aparezca vacía o falten elementos. Para visualizar correctamente el funcionamiento del EKF, debes configurar manualmente los *Displays* y el marco de referencia.

1. Configuración del Entorno (Global Options)

- **Fixed Frame:** Debe estar establecido estrictamente en `map`. *Advertencia: Si este valor está en `world` o `base_link`, no verás la posición global del robot en el mapa.*

2. Elementos a Visualizar (Displays) Asegúrate de agregar (botón **Add**) y configurar los siguientes *Topics*:

1. **Estimación EKF (Flecha Azul)**
   Representa la pose calculada por tu algoritmo.
   - **Topic:** `/NOMBRE_ROBOT/ekf_localization_node/pose`
   - **Tipo:** `nav_msgs/Odometry`
   - **Configuración:** Cambia el color de la flecha a **Azul** y habilita la visualización de la **Covarianza** (elipse) para ver la incertidumbre.
2. **Ground Truth (Flecha Roja)**
   Muestra la posición real exacta del robot (solo disponible en simulación/Matrix).
   - **Topic:** `/NOMBRE_ROBOT/duckiematrix_interface_node/state`
   - **Configuración:** Cambia el color a **Rojo** para distinguirlo de tu estimación.
3. **Landmarks (Marcadores)**
   Muestra las señales de tráfico (AprilTags) en el mapa.
   - **Topic:** `/NOMBRE_ROBOT/map_markers`
   - **Tipo:** `visualization_msgs/MarkerArray`
   - **Significado de Colores:**
     - **Esferas Verdes:** Landmark cargado en el mapa pero **no** detectado actualmente.

- **Esferas Azules:** Landmark siendo detectado por la cámara en este instante (se usa para corregir la posición).

Resumen de Diagnóstico: Si no ves nada, revisa: 1) Que el **Fixed Frame** sea `map`. 2) Que el robot (virtual o real) esté publicando datos (usa `rostopic list` para verificar).

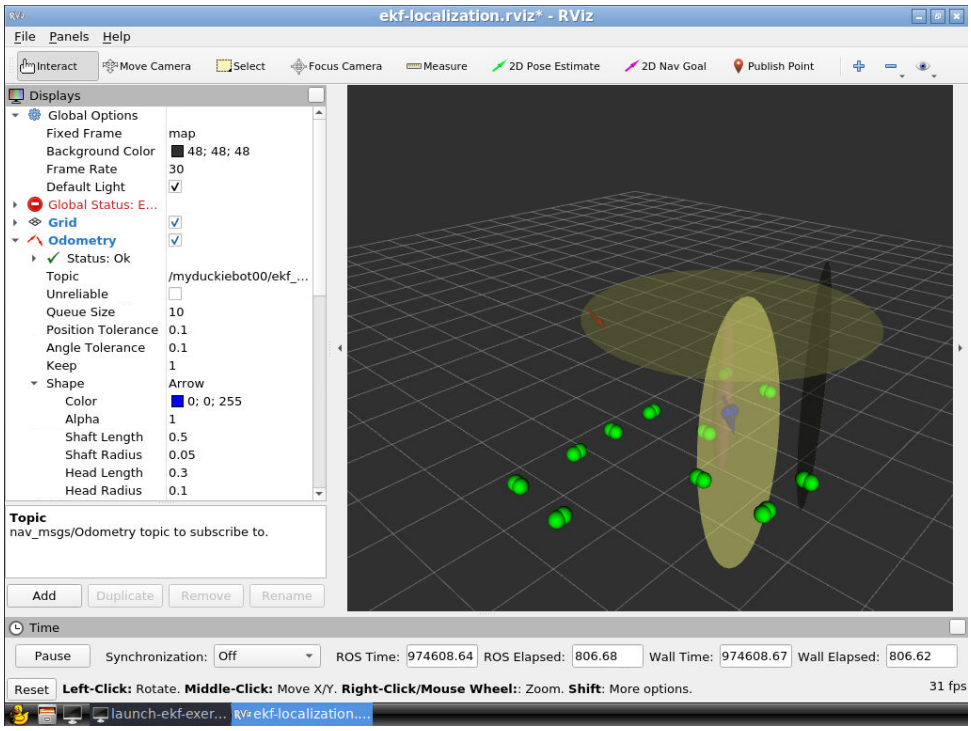

Figura 15: Visualización del EKF en RViz. **Haz clic sobre la imagen** para ver un video demostrativo de la simulación en funcionamiento en YouTube.