# Control del Sistema de Comunicación Electrónico para la Adquisición de Imágenes en el Entorno Simulado

## Introducción

Este documento presenta una descripción detallada del sistema de comunicación electrónico desarrollado para la adquisición de imágenes en un entorno simulado. Contiene información sobre el modelado C4 del sistema. Además, se incluyen conclusiones y recomendaciones derivadas del trabajo realizado.

## Modelado C4 del Sistema Terminal

El modelado C4 describe los componentes principales del sistema desde diferentes niveles de abstracción:

### Nivel Contexto: Interacción del Sistema Terminal con Actores y Sistemas Externos

El **Sistema Terminal** es el software central que permite a los operadores controlar y monitorear el **Pipeline Inspection Robot**. Además, interactúa con varios sistemas externos para garantizar la coordinación y funcionalidad completa del sistema en la Figura 1 se muestra el diagrama del sistema de contexto.

**Actores y Sistemas Interactuantes**

1. **Operator (Operador)**
    - **Descripción**: Persona técnica encargada de operar el sistema de inspección de tuberías.
    - **Interacción**:
        - El operador utiliza el **Sistema Terminal** como interfaz principal para controlar el robot.
        - Realiza tareas como:
            - Enviar comandos para el movimiento del robot.
            - Monitorear los datos de inspección en tiempo real, incluidos los flujos de video y telemetría.
2. **Pipeline Inspection Robot**
    - **Descripción**: Robot físico diseñado para realizar inspecciones en tuberías.
    - **Interacción**:
        - El **Sistema Terminal** envía comandos de control al robot, incluyendo:
            - Movimiento (desplazamiento en el entorno de inspección).
            - Posicionamiento de la cámara (para capturar imágenes y video).
        - El robot, a su vez, transmite datos de telemetría (estado del sistema, sensores, posición) al **Sistema Terminal**.
3. **DVR System**

- **Descripción**: Sistema de grabación de video digital (DVR) que administra la grabación, almacenamiento y reproducción de los flujos de video capturados por el robot.
- **Interacción**:
  - El **Sistema Terminal** controla las operaciones del DVR, como:
    - Iniciar y detener grabaciones.
    - Recuperar flujos de video en vivo para su monitoreo.
    - Acceder a grabaciones almacenadas para análisis posterior.
  - Este sistema actúa como intermediario para almacenar y proporcionar acceso a las grabaciones.

4. **Panel and Feeder System**
   - **Descripción**: Sistema externo encargado de controlar el panel y las operaciones de alimentación del robot. También permite enviar acciones del usuario y recibir comandos del joystick.
   - **Interacción**:
     - El **Sistema Terminal** se comunica con el **Panel and Feeder System** a través de comunicación serie.
     - Funciones principales:
       - Controlar el movimiento del robot.
       - Gestionar la orientación y posicionamiento de la cámara.
       - Coordinar las operaciones del sistema de alimentación.

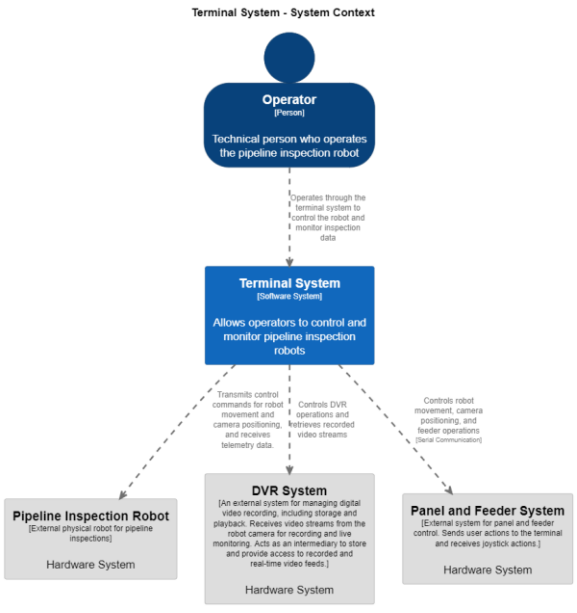


Figura 1 Terminal System- Diagrama de contexto

## Nivel Contenedor: Detalle de los Componentes del Sistema Terminal

El nivel contenedor describe cómo se descompone el **Sistema Terminal** en módulos principales (contenedores), sus roles, y cómo interactúan entre sí y con los sistemas externos. Este enfoque ayuda a comprender las responsabilidades de cada módulo dentro del sistema.

### 1. Sistema Terminal

El **Sistema Terminal** es un sistema de software que permite al operador controlar y monitorear el **Pipeline Inspection Robot**.

Está compuesto por los siguientes contenedores que se muestran en la Figura 2:

1. **Communication Module**
    - **Descripción**: Este contenedor gestiona la comunicación con el robot mediante el protocolo CAN.
    - **Tecnologías**: Python + python-can.
    - **Responsabilidades**:
        - Envía comandos para controlar los movimientos del robot y la cámara.
        - Recibe datos de telemetría del robot, incluyendo estado de los sensores y posición.
    - **Interacción**:
        - **Con el Robot**: Se conecta directamente al **Pipeline Inspection Robot** a través de CAN.
        - **Con el Inspection Module**: Recibe comandos de movimiento y envía datos de telemetría.
2. **Video Module**
    - **Descripción**: Maneja el streaming, grabación y almacenamiento de video, además de la gestión de sesiones de inspección.
    - **Tecnologías**: Python + OpenCV + PyQt.
    - **Responsabilidades**:
        - Transmite flujos de video en tiempo real desde el robot.
        - Permite grabar y almacenar los flujos de video.
        - Gestiona las sesiones de video en coordinación con el **Inspection Module**.
    - **Interacción**:
        - **Con el DVR System**: Utiliza el protocolo RTSP para transmitir flujos de video en tiempo real.
        - **Con el Inspection Module**: Recibe configuraciones y envía flujos de video en tiempo real.
3. **Panel and Feeder Module**
    - **Descripción**: Controla la interfaz del panel y los mecanismos de alimentación asociados al robot.
    - **Tecnologías**: Python + PySerial.
    - **Responsabilidades**:
        - Recibe señales desde el panel y el sistema de alimentación.

        - Controla las operaciones relacionadas con el panel.
    - **Interacción**:
        - **Con el Panel and Feeder System**: Usa comunicación serial para enviar y recibir señales.
        - **Con el Inspection Module**: Publica eventos relacionados con el panel y las operaciones de alimentación.
4. **Inspection Module**
    - **Descripción**: Es el módulo principal del sistema, encargado de coordinar la interacción entre los otros contenedores.
    - **Tecnologías**: PyQt.
    - **Responsabilidades**:
        - Proporciona la interfaz gráfica al operador para controlar el robot.
        - Coordina la comunicación entre los módulos de video, comunicación y alimentación.
        - Configura sesiones en el **Video Module** y gestiona los comandos enviados al robot.
    - **Interacción**:
        - **Con el Operator**: Proporciona una GUI para enviar comandos y monitorear datos.
        - **Con los demás módulos**: Publica y suscribe señales para coordinar la funcionalidad del sistema.

**2. Sistemas Externos**

Los siguientes sistemas externos interactúan con el Sistema Terminal:

1. **Pipeline Inspection Robot**
    - Controlado por el **Communication Module**.
    - Proporciona telemetría y recibe comandos de movimiento y posicionamiento.
2. **DVR System**
    - Gestionado por el **Video Module**.
    - Recibe flujos de video a través de RTSP para grabación y reproducción.
3. **Panel and Feeder System**
    - Conectado al **Panel and Feeder Module**.
    - Envía señales de usuario y recibe comandos relacionados con el panel y alimentación.

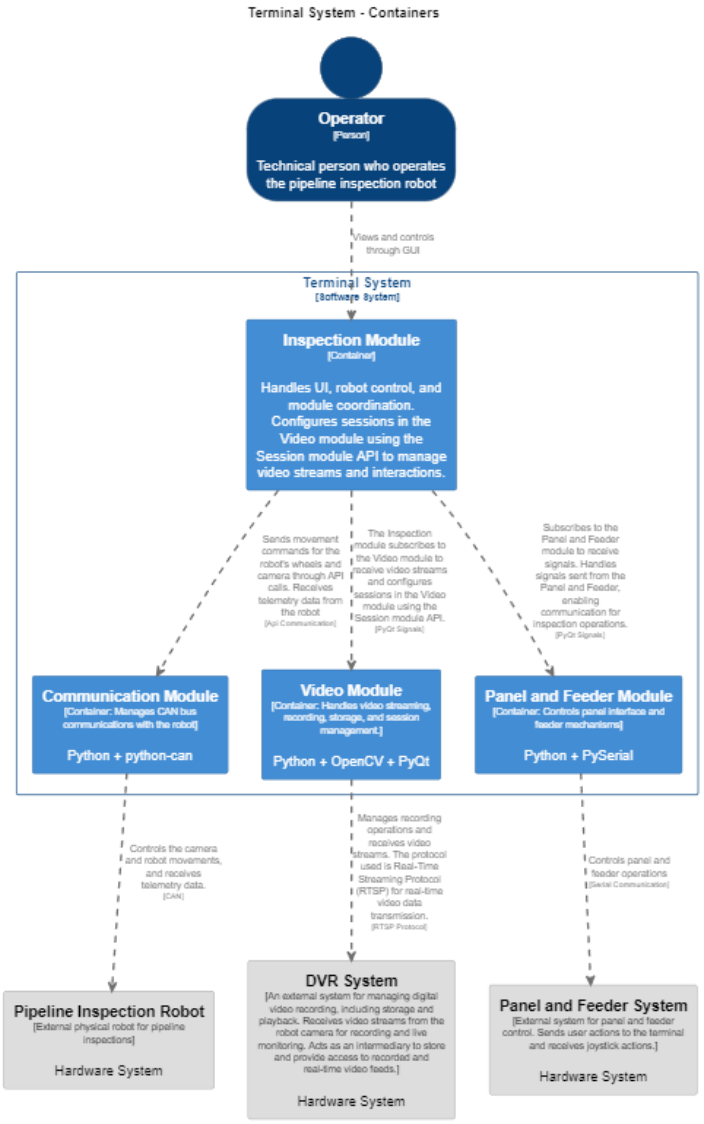


**Figura 2 Nivel de contenedores del sistema terminal**

## *Nivel Componente: Detalle de los Módulos Internos del Sistema Terminal*

El nivel de componentes descompone cada contenedor en sus principales submódulos y muestra cómo se conectan entre sí para realizar las funciones específicas del sistema. Este nivel es clave para entender la implementación detallada de cada funcionalidad.

### Componentes Principales del Módulo de Inspección

En la figura 3 se muestran los componentes del Módulo de inspección y como se relacionan.

- **MainWindow**

- **Descripción**: Sirve como la interfaz gráfica de usuario (GUI) para el Módulo de Inspección.
- **Responsabilidades**:
  - Mostrar datos en tiempo real, incluyendo:
    - Actualizaciones del Feeder
    - Telemetría del robot.
    - Flujos de video.
    - Estado de las sesiones activas.
  - Facilitar la interacción del operador con todos los servicios del sistema a través de señales PyQt.
- **Interacciones**:
  - Recibe datos desde todos los demás componentes del Módulo de Inspección.
  - Proporciona al operador una experiencia centralizada y accesible.

**FeederUpdateService**

- **Descripción**: Gestiona las actualizaciones y la comunicación con el sistema del Feeder
- **Responsabilidades**:
  - Enviar comandos al sistema de alimentación a través de FeederControllerPort.
  - Notificar estados y eventos a la GUI mediante GuiFeederObserverAdapter.
- **Interacciones**:
  - Coordina con el sistema de alimentación.
  - Publica actualizaciones en la GUI.

**TelemetryUpdateService**

- **Descripción**: Procesa las actualizaciones de telemetría recibidas del robot.
- **Responsabilidades**:
  - Decodificar y estructurar los datos de telemetría.
  - Notificar eventos a través de TelemetryObserverPort.
  - Actualizar la GUI utilizando GuiTelemetryObserverAdapter.
- **Interacciones**:
  - Recibe datos de telemetría en tiempo real del robot.
  - Envia actualizaciones al operador a través de la GUI.

**VideoUpdateService**

- **Descripción**: Gestiona la transmisión de video en tiempo real.
- **Responsabilidades**:
  - Subscribirse a los flujos de video proporcionados por el Módulo de Video.

- Enviar actualizaciones de video a la GUI utilizando GuiVideoObserverAdapter.
- **Interacciones**:
  - Trabaja directamente con el Módulo de Video para obtener y mostrar flujos.

### SessionServices

- **Descripción**: Gestiona las operaciones relacionadas con las sesiones del sistema.
- **Responsabilidades**:
  - Iniciar, detener y descargar sesiones de inspección.
  - Coordinar con SessionControllerPort para administrar las sesiones.
- **Interacciones**:
  - Recibe comandos del operador a través de la GUI.
  - Envia información sobre el estado de las sesiones.

### PanelUpdateServices

- **Descripción**: Maneja las actualizaciones relacionadas con el control del robot y la cámara.
- **Responsabilidades**:
  - Controlar el movimiento del robot a través de MovementControllerPort.
  - Gestionar las operaciones de la cámara mediante CameraControllerPort.
  - Enviar y recibir actualizaciones relacionadas con estas operaciones a la GUI utilizando PanelUpdateServicesPort.
- **Interacciones**:
  - Publica eventos relacionados con el robot y la cámara en la GUI.
  - Coordina señales entre el operador y el sistema de panel y alimentación.

Figura 3 Componentes del Módulo de inspección

## *Nivel Componentes: Detalle del Módulo de Comunicación*

El **Módulo de Comunicación** es responsable de gestionar las interacciones entre el sistema terminal y el robot de inspección mediante el bus CAN. Este módulo actúa como un puente entre el Módulo de Inspección y el hardware del robot, facilitando el control de sus funciones principales. En la figura 4 se muestra el diagrama del modulo de comunicación.

**Componentes Principales del Módulo de Comunicación**

1. **TelemetryServices**
    - **Descripción**: Maneja los datos de telemetría transmitidos por el robot a través del bus CAN.
    - **Responsabilidades**:
        - Leer y decodificar los datos de telemetría del robot.
        - Enviar actualizaciones de telemetría a la GUI mediante el adaptador PyqtTelemetryObserverAdapter.
        - Proporcionar datos como posición, estado de sensores y condiciones operativas en tiempo real.
    - **Interacciones**:
        - Lee datos de telemetría desde el componente **CAN Bus**.
        - Publica las actualizaciones en el **Módulo de Inspección**.
2. **CameraServices**
    - **Descripción**: Controla las operaciones de la cámara del robot.
    - **Responsabilidades**:
        - Enviar comandos para posicionar y operar la cámara a través del adaptador CanCameraControllerAdapter.
        - Proporcionar funciones como zoom, enfoque y activación de la transmisión de video.
    - **Interacciones**:
        - Usa el **CAN Bus** para enviar comandos de cámara al robot.
        - Recibe comandos del **Módulo de Inspección** para operaciones de la cámara.
3. **MovementService**
    - **Descripción**: Gestiona los movimientos del robot, incluyendo desplazamientos y maniobras.
    - **Responsabilidades**:
        - Enviar comandos de movimiento a las ruedas del robot mediante el adaptador CanWheelsControllerAdapter.

        - Coordinar movimientos suaves y precisos para inspección eficiente.
    - **Interacciones**:
        - Envia comandos de movimiento al robot a través del **CAN Bus**.
        - Recibe comandos del **Módulo de Inspección**.
4. **CAN Bus**
    - **Descripción**: Es el canal de comunicación compartido por todos los componentes del Módulo de Comunicación.
    - **Responsabilidades**:
        - Transmitir y recibir tramas CAN entre el sistema terminal y el robot.
        - Actuar como un enlace confiable para las comunicaciones de control y telemetría.
    - **Interacciones**:
        - Envia comandos desde **MovementService** y **CameraServices** hacia el robot.
        - Recibe datos de telemetría y responde a solicitudes del robot.

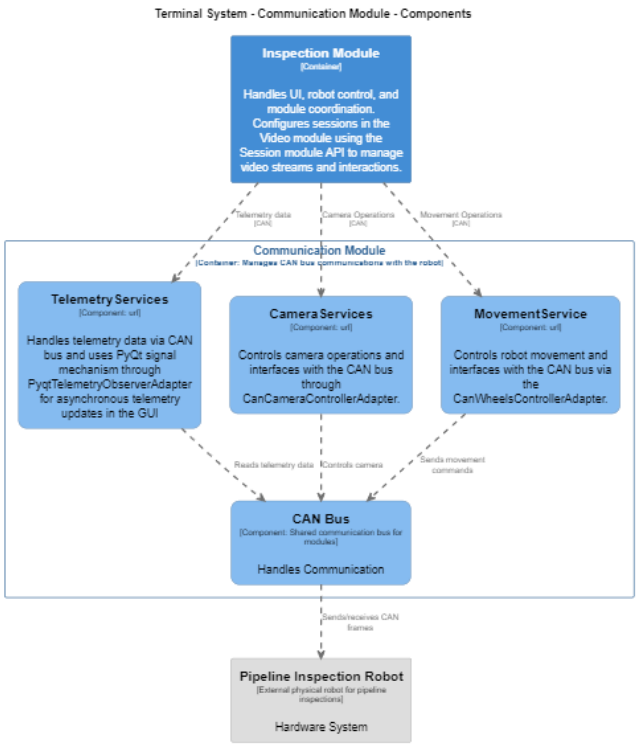


Figura 4 Componentes del Módulo de comunicación.

## *Nivel Componentes: Detalle del Módulo de Panel y Feeder*

El Módulo de Panel y Feeder es responsable de gestionar las interacciones entre el sistema terminal y el sistema externo de panel y alimentación. Este módulo garantiza el control y sincronización eficiente de las operaciones relacionadas con el panel de control y los mecanismos de alimentación. En la figura 5 se muestran los componentes del modulo del Panel y el Feeder.

**Componentes Principales del Módulo de Panel y Feeder**

1. **Panel Services**
    - **Descripción**: Este componente implementa las operaciones relacionadas con el panel de control del robot.
    - **Responsabilidades**:
        - Controlar y gestionar las acciones del panel.
        - Enviar actualizaciones a la GUI utilizando señales PyQt.
        - Utilizar PanelAndFeederControllerPort para comunicarse con el sistema externo.
        - Implementar PanelObserverPort para notificar actualizaciones relevantes.
    - **Interacciones**:
        - Envía comandos al **Serial Panel Feeder Adapter** para controlar el panel.
        - Recibe comandos del **Módulo de Inspección** para ejecutar operaciones de panel.
        - Publica actualizaciones en la GUI.
2. **Feeder Services**
    - **Descripción**: Este componente implementa las operaciones relacionadas con el sistema de alimentación.
    - **Responsabilidades**:
        - Coordinar los mecanismos de alimentación del robot.
        - Enviar actualizaciones a la GUI mediante señales PyQt.
        - Utilizar PanelAndFeederControllerPort para comunicarse con el sistema externo.
        - Implementar FeederObserverPort para proporcionar notificaciones en tiempo real.
    - Interacciones:
        - Envía comandos al Serial Panel Feeder Adapter para controlar el sistema de alimentación.
        - Recibe comandos del Módulo de Inspección para ejecutar operaciones de alimentación.
        - Publica actualizaciones en la GUI.
3. Serial Panel Feeder Adapter

- Descripción: Actúa como un adaptador que implementa la interfaz PanelAndFeederControllerPort utilizando la biblioteca PySerial.
- Responsabilidades:
  - Traducir los comandos de control del panel y alimentación en datos seriales.
  - Establecer y mantener la comunicación con el sistema externo de panel y alimentación.
- **Interacciones**:
  - Recibe comandos de los componentes **Panel Services** y **Feeder Services**.
  - Transmite comandos al sistema externo de panel y alimentación.
  - Proporciona un enlace confiable entre el sistema terminal y el hardware externo.

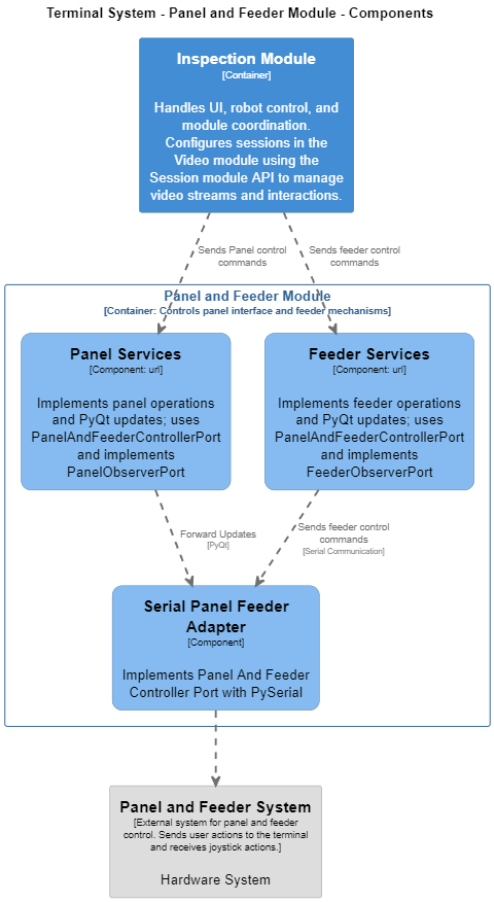


Figura 5 Componentes del módulo de Panel y Feeder

## *Nivel Componentes: Detalle del Módulo de Video*

El **Módulo de Video** es responsable de gestionar las operaciones relacionadas con el video, como la transmisión, grabación, almacenamiento y administración de sesiones. Este módulo garantiza que los flujos de video se transmitan y graben eficientemente mientras permite al operador acceder a ellos en tiempo real. En la figura 6 se pueden ver los componentes del modulo de video.

**Componentes Principales del Módulo de Video**

1. **Session Services**
    - **Descripción**: Administra las sesiones de grabación de video, incluyendo la creación, descarga y recuperación de datos relacionados con las sesiones.
    - **Responsabilidades**:
        - Configurar sesiones de grabación de video mediante Session module API.
        - Conectar con sistemas DVR a través de DvrControllerPort, implementado por HikvisionDvrControllerAdapter.
        - Gestionar almacenamiento local y metadatos utilizando RepositoryPort, implementado por TinyDBRepositoryAdapter.
    - **Interacciones**:
        - Coordina con el **DVR System** para gestionar sesiones de grabación y reproducción.
        - Recibe configuraciones del **Módulo de Inspección** para iniciar y detener sesiones.
        - Almacena metadatos en un repositorio local para facilitar la recuperación de sesiones.
2. **Video Services**
    - **Descripción**: Gestiona la transmisión de video y las notificaciones en tiempo real a los observadores.
    - **Responsabilidades**:
        - Capturar y transmitir flujos de video desde las fuentes utilizando VideoControllerPort.
        - Notificar actualizaciones en tiempo real mediante VideoObserverPort, implementado por PyQtVideoObserverAdapter y PyQtSignalCamera.
        - Facilitar la visualización de video en la GUI del operador.
    - **Interacciones**:
        - Coordina con el **DVR System** para la transmisión de video en tiempo real.
        - Notifica al **Módulo de Inspección** sobre actualizaciones de video para que sean reflejadas en la interfaz gráfica.

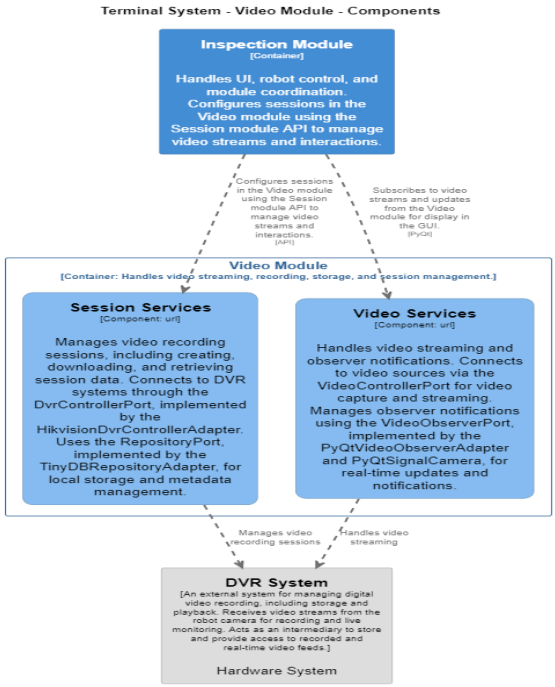


Figura 6 Componentes del módulo de video

## Arduino Due - Panel de Control

### Descripción General

El programa desarrollado para el Arduino Due actúa como el controlador principal del panel de control del robot. Su propósito es interpretar señales provenientes de diferentes módulos, tales como el encoder, joysticks, control de iluminación, y sensores de posición, para realizar acciones específicas. Además, el programa envía y recibe datos que permiten el monitoreo y control del sistema en tiempo real.

### Implementación Actual

El código está dividido en varias secciones clave que gestionan los diferentes componentes del sistema. A continuación, se describen los principales módulos:

1. **Encoder**
    - Pines configurados: encoderPinA, encoderPinB.
    - Registra el conteo de pulsos y envía mensajes de estado a intervalos definidos.
    - Incluye funcionalidad de reinicio mediante un botón físico.
2. **Joysticks**
    - Controlan el movimiento del robot (adelante, atrás, izquierda, derecha) y el movimiento de la cámara (inclinación y paneo).

    - Pines asignados para cada dirección de control: Backward, Forward, Left, Right para el robot y TiltUp, TiltDown, PanLeft, PanRight para la cámara.
3. **Control de Iluminación**
    - Utiliza un potenciómetro conectado al pin analógico Light para ajustar la intensidad de la luz.
4. **Sensores de Límite**
    - Señal del sensor de fin de carrera del alimentador conectada al pin FeederButton.
5. **Gestión de Señales**
    - Implementa lógica de debounce para filtrar señales ruidosas de botones y joysticks.
    - Usa variables y temporizadores para gestionar el procesamiento de señales.

**Diagrama de Flujo**

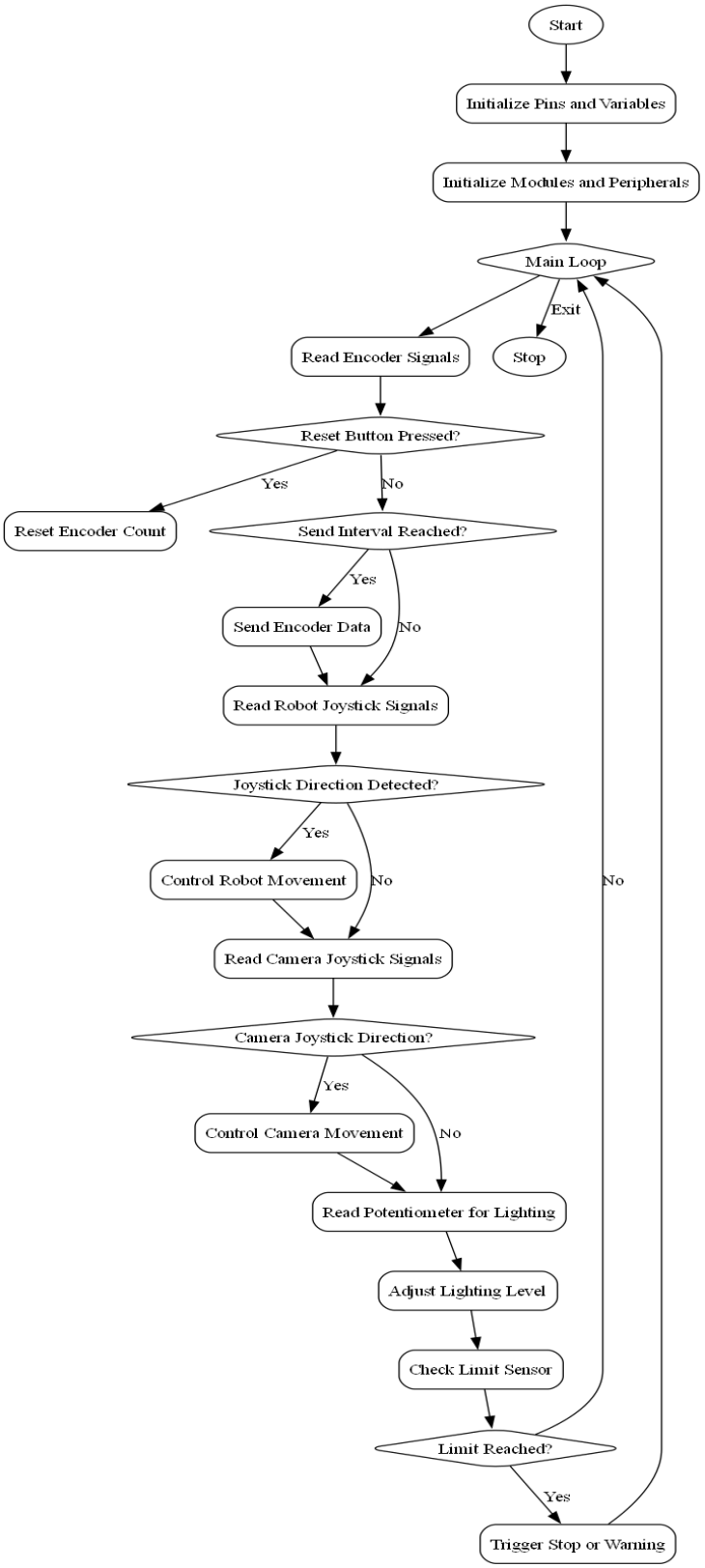


Figura 7 Diagrama de flujo del programa ArduinoDue

**Explicación del Diagrama de Flujo**

El programa desarrollado para el **Arduino Due** implementa la lógica necesaria para controlar y monitorear los módulos del robot, como el encoder, los joysticks y el potenciómetro. A continuación, se describe cada sección del diagrama de flujo, que refleja la lógica y las interacciones del sistema.

**1. Inicialización del Sistema**

El programa comienza configurando los pines y variables requeridos:

- **Encoder**: Pines digitales encoderPinA (2) y encoderPinB (3) para recibir pulsos.
- **Botón de Reinicio del Encoder**: Pin digital ResetButton (5).
- **Joysticks**:
    - Para el robot: Pines digitales Forward (10), Backward (13), Left (12), Right (11).
    - Para la cámara: Pines digitales TiltUp (6), TiltDown (9), PanLeft (8), PanRight (7).
- **Control de Enfoque**: Pines digitales FocusIn (36) y FocusOut (40).
- **Potenciómetro de Iluminación**: Entrada analógica Light (A1).
- **Sensor de Fin de Carrera**: Pin digital FeederButton (4).

Además, se inicializan variables para manejar temporizadores (lastPrintTime y printInterval) y filtros de debounce.

**2. Ciclo Principal**

El bucle principal procesa las señales de entrada y ejecuta las acciones correspondientes. Las tareas se realizan en el siguiente orden:

1. **Lectura del Encoder**:
    - Se procesan los pulsos recibidos por los pines del encoder mediante interrupciones. Si el botón de reinicio es presionado, se verifica con debounce y se reinicia el contador (pulseCount).
    - Cada segundo (definido por printInterval), los datos del encoder son enviados al sistema central.
2. **Control del Robot con Joysticks**:
    - Las señales de los joysticks son leídas de los pines digitales asignados (Forward, Backward, Left, Right).
    - Según las señales detectadas, se ejecutan comandos para mover las ruedas del robot en la dirección indicada.
3. **Control de la Cámara con Joysticks**:

    - Las señales del joystick dedicado a la cámara (TiltUp, TiltDown, PanLeft, PanRight) son leídas y procesadas para ajustar su inclinación y rotación.
4. **Ajuste de Iluminación**:
    - El valor analógico del potenciómetro conectado al pin Light (A1) es leído y traducido a un comando que ajusta la intensidad de la luz.
5. **Detección de Eventos del Sensor de Fin de Carrera**:
    - Se verifica el estado del pin FeederButton. Si se detecta que el sensor ha alcanzado un límite, se ejecuta una acción de detención o advertencia.

### 3. Gestión de Interrupciones

- **Interrupciones del Encoder**: Las señales de los pines encoderPinA y encoderPinB se manejan mediante interrupciones para asegurar un conteo preciso de los pulsos sin afectar el rendimiento general del sistema.
- Reinicio del Encoder: Si el botón ResetButton es presionado, se utiliza un temporizador para filtrar señales ruidosas y reiniciar el conteo del encoder.

### 4. Envío de Datos

- Los datos del encoder, joysticks, potenciómetro y sensor de límite se envían al sistema central cada segundo, garantizando un monitoreo en tiempo real.

## ESP32 - Robot de Inspección

**Descripción General**
Este programa está diseñado para controlar un robot de inspección basado en el microcontrolador ESP32. El sistema integra múltiples módulos para realizar tareas como la comunicación con periféricos (telemetría, control de cámaras, y movimiento de ruedas) y la recolección de datos de sensores ambientales y de orientación. El robot está equipado con un sistema de comunicaciones SPI a CAN para interactuar con otros dispositivos en la red, y utiliza módulos como el BME680 para mediciones ambientales y el MPU9250 para datos de movimiento e inclinación.

**Implementación Actual**

**Comunicación SPI a CAN**

El sistema incluye un módulo MCP2515 para la comunicación CAN, configurado mediante SPI. Este módulo gestiona los mensajes enviados y recibidos para sincronizar las operaciones del robot.

- **Buffer de Recepción**: Se utiliza un arreglo rxBuf para almacenar los datos recibidos.
- **Id de Mensajes**: Los mensajes CAN son identificados por un ID único rxId para organizar las respuestas y acciones.

**Control de Sensores**

- **BME680**: Recoge datos de temperatura, humedad, presión y calidad del aire. Utiliza la biblioteca Adafruit_BME680 para configuración y lecturas.
- **MPU9250**: Proporciona datos de aceleración, giroscopio y magnetómetro para calcular la orientación y el movimiento.

**Control de Cámara**

El robot utiliza comandos TTL a través de RS485 para controlar movimientos de la cámara y ajustes de iluminación:

- **Inicialización**: El comando camero_init inicializa el módulo de la cámara.
- **Control de Movimientos**: Un array camera_control almacena comandos para manejar el movimiento y la luz.

**Interrupciones**

Se utiliza un temporizador basado en hardware para enviar mensajes periódicamente:

- **Timer0**: Configurado para generar interrupciones y actualizar la variable control_mensajes, que regula la frecuencia de envío de comandos.

**Comunicación Serie**

La configuración de pines define los canales serie:

- **RS485**: Se conecta a través de TXD1 (pin 17) y RXD1 (pin 16).

**Diagrama de Flujo**

Presentación del diagrama de flujo del programa con una explicación clara de cada etapa del proceso.

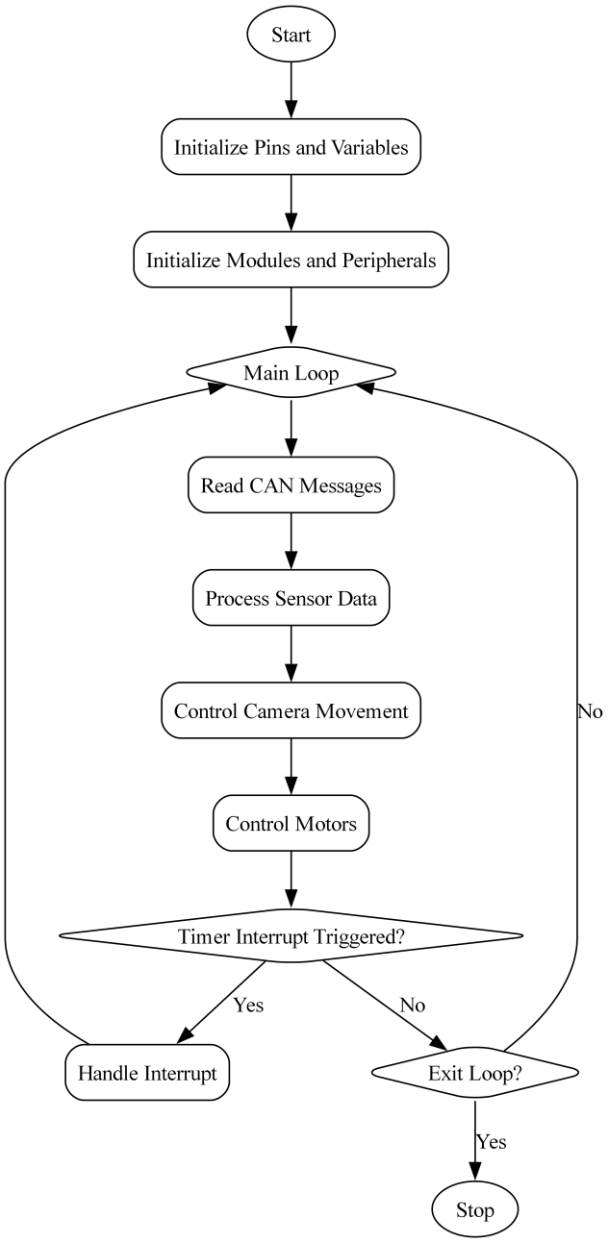


**Figura 8 Diagrama de Flujo del programa del ESP32 Robot de Inspección.**

### Explicación del Diagrama de Flujo

El diagrama de flujo refleja cómo se estructuran y procesan las tareas clave, desde la inicialización hasta la ejecución de las acciones principales. A continuación, se explica cada sección en detalle:

## 1. Inicialización del Sistema

El programa comienza configurando todos los periféricos y variables necesarias. En esta etapa, se aseguran las conexiones y el funcionamiento básico de los módulos del sistema:

- **Módulo CAN**: Inicialización del bus SPI para recibir mensajes y comunicarse con otros módulos del sistema.
- **Sensores**: Configuración de sensores como temperatura, humedad y posición (IMU), asegurando que las mediciones estén listas para ser procesadas.
- **Cámara y Motores**: Preparación para controlar los movimientos de la cámara y los motores, utilizando pines digitales dedicados.
- **Interrupciones**: Configuración del temporizador para generar interrupciones periódicas que permitan sincronizar tareas críticas del sistema.

El sistema queda listo para entrar en el ciclo principal.

## 2. Ciclo Principal

En este bucle, el programa ejecuta de forma continua las tareas asignadas. Estas tareas incluyen la lectura de datos, el procesamiento de señales, y el control de actuadores.

### 2.1. Procesamiento de Mensajes CAN

- Los mensajes recibidos a través del bus CAN son leídos y procesados. Estos mensajes pueden contener comandos para los motores, ajustes para la cámara, o configuraciones de sensores.
- Los datos recibidos se almacenan en búferes temporales para su análisis.

### 2.2. Lectura y Procesamiento de Sensores

- Se recopilan datos de sensores como temperatura, presión, velocidad del viento, y humedad.
- Los valores leídos se procesan para eliminar ruido (debounce o filtros digitales) y prepararse para su envío o uso en el control del sistema.

### 2.3. Control de Motores

- En función de los datos procesados y las entradas recibidas, se generan comandos para ajustar la velocidad y dirección de los motores. Esto incluye:
    - **Movimiento del robot**: Basado en datos de joysticks o comandos remotos.
    - **Parada o cambio de dirección**: En respuesta a eventos como la detección de límites por sensores.

### 2.4. Control de la Cámara.

- Los comandos recibidos para la cámara ajustan su inclinación (tilt) y rotación (pan). Los valores se procesan para garantizar movimientos suaves y precisos.

**3. Gestión de Interrupciones**

- El sistema utiliza interrupciones periódicas para tareas críticas que no pueden depender del ciclo principal.
- **Interrupciones Temporizadas**: Estas interrupciones sincronizan eventos importantes, como el envío periódico de datos o el manejo de mensajes CAN.
- **Interrupciones del Encoder**: Garantizan que las señales del encoder sean procesadas en tiempo real, asegurando mediciones precisas.

**4. Envío de Datos**

- A intervalos regulares (por ejemplo, cada segundo), los datos recopilados de sensores, encoders, y actuadores se envían al sistema central.
- Este proceso garantiza que los operadores tengan acceso en tiempo real a la información del sistema

**5. Manejo de Eventos Críticos**

- **Detección de Límites**: Si un sensor detecta que se ha alcanzado un límite, el sistema ejecuta una acción de seguridad, como detener motores o activar una advertencia.
- **Reinicio de Contadores**: Botones dedicados permiten reiniciar contadores como los del encoder.

## Conclusiones y Recomendaciones

**Conclusiones**

1. **Eficiencia del Sistema**: El sistema desarrollado demuestra una alta capacidad para gestionar y coordinar múltiples módulos, garantizando el control efectivo del Pipeline Inspection Robot.

2. **Modularidad y Escalabilidad**: La implementación modular permite futuras expansiones y modificaciones del sistema, mejorando su adaptabilidad a nuevos escenarios.

3. **Integración de Tecnologías**: El uso de tecnologías como CAN Bus, RTSP, y PyQt asegura una comunicación robusta entre los módulos y un monitoreo en tiempo real de las operaciones.

4. **Interfaz de Usuario Intuitiva**: La GUI facilita la interacción del operador con el robot, centralizando el control y monitoreo en una sola plataforma.

**Recomendaciones**

1. **Manejo de Iluminación desde la Interfaz de Usuario**: Implementar el control de la iluminación directamente en la interfaz de usuario en lugar de depender de un potenciómetro físico. Esto asegurará un uso más preciso y eficiente del sistema, además de mejorar la experiencia del operador.
2. **Gestión de Videos por Lotes:** Permitir que el sistema descargue videos por lotes para facilitar la transferencia y manejo de grandes volúmenes de grabaciones, optimizando los flujos de trabajo.

3. **Eliminación de Grabaciones:** Incorporar una funcionalidad para eliminar grabaciones de las sesiones directamente desde la interfaz de usuario, asegurando una gestión eficiente del almacenamiento.