EPC Lector USB

Este directorio contiene un ejemplo de cómo leer eventos de dispositivos USB (como teclados y lectores de códigos de barras) conectados a un PLC Phoenix Contact EPC 1502 utilizando Yocto Linux Kirkstone. Se utiliza un script en Python que se ejecuta desde Node-RED para interpretar los eventos del puerto USB.

> **i Nota**: Este proyecto está diseñado y probado específicamente para el PLC Phoenix Contact EPC 1502 con Yocto Linux Kirkstone. Puede requerir ajustes para otros modelos o sistemas.

Tabla de Contenidos

- Introducción
- Contenido del Directorio
- Instalación
  - Configuración del Entorno
  - Preparación del Script en Python
  - Modificación de Node-RED
- Uso del Proyecto
- Cómo Funciona el Código
- Ejemplos de Uso
- Programa en Node-RED
  - Ejemplo de Uso del Flujo
- Instrucciones de Configuración Adicionales
- Licencia

Introducción

El objetivo de este ejemplo es demostrar cómo leer e interpretar los eventos generados por dispositivos USB conectados al PLC, como un teclado o un lector de códigos de barras, en un entorno Node-RED.

Contenido del Directorio

La estructura del directorio es la siguiente:

```
EPCLectorUSB/
├── src/
│   └── usbdrivers.py             # Script en Python para leer eventos del puerto
USB
├── flows/
│   └── flows_usb.json            # Flujos para importar en Node-RED
└── README.md                     # Documentación principal del proyecto
```

Instalación

Configuración del Entorno

1. **Configurar acceso root:**
   - Crear nueva contraseña para el usuario `root` en el dispositivo:

```
sudo passwd root
```

> ⚠ **Advertencia**: Cambiar la contraseña de root es una operación sensible; asegúrate de recordar la nueva contraseña para evitar problemas de acceso al dispositivo.

Preparación del Script en Python

1. Crear un directorio para el script:

```
mkdir /home/root/usbdrivers
```

2. Crear el archivo `usbdrivers.py`:

```
touch /home/root/usbdrivers/usbdrivers.py
```

3. Copiar el código Python del archivo `src/usbdrivers.py` a este directorio en el dispositivo:

```
nano /home/root/usbdrivers/usbdrivers.py
```

(Pegar el contenido y guardar).

Modificación de Node-RED

1. Abrir el archivo de configuración de Node-RED:

```
nano /opt/plcnext/appshome/data/60002172000551/docker-compose.yml
```

> **i Nota**: El directorio `60002172000551` puede variar según la versión de Node-RED instalada; consulta la estructura de archivos en el dispositivo si es necesario.

2. Modificar el archivo de la siguiente manera:

```
version: "3.7"
services:
  node-red:
    image: ${IMAGE_NAME}:${IMAGE_TAG}
    ports:
      - 51880:1880
```

```
    user: root
    volumes:
      - ./volumes/node-red:/data
      - /home/root/usbdrivers:/usbdrivers
    restart: unless-stopped
    privileged: true
```

> ❗ **Importante**: Verifica que la ruta `/home/root/usbdrivers` esté correctamente escrita y montada. Los permisos incorrectos pueden impedir la lectura de eventos USB.

3. Reiniciar el dispositivo para aplicar los cambios:

```
reboot
```

Uso del Proyecto

Con Node-RED configurado para utilizar el script Python, se puede conectar un teclado o lector de código de barras al PLC y observar cómo se interpretan los eventos en tiempo real.

> ☑ **Tip**: Conecta y prueba el dispositivo USB (teclado o lector de código de barras) antes de comenzar. Puedes verificar la salida de eventos USB en Node-RED para asegurarte de que el script está funcionando.

Cómo Funciona el Código

Flujo General del Código

1. **Lectura del Evento**:
   - El script abre el archivo `/dev/input/eventX`, donde `X` representa el número del evento USB asignado al dispositivo.
   - Lee paquetes de 24 bytes que contienen información sobre el evento (timestamp, tipo, código y valor).
2. **Interpretación del Evento**:
   - Los eventos de tipo `1` (evento de botón) con un valor `1` (presionado) representan pulsaciones de teclas.
   - El script convierte el código de tecla (`code`) usando el diccionario `key_map` para traducirlo a caracteres legibles.
3. **Buffer y Envío de Datos**:
   - Los caracteres se almacenan en un buffer temporal.
   - Cuando el código detecta la tecla Enter (código `28`), imprime el contenido del buffer y lo vacía para la siguiente secuencia de entrada.

Integración con Node-RED

Node-RED se utiliza para ejecutar el script `usbdrivers.py` y mostrar los datos de eventos USB capturados en tiempo real. La integración con Node-RED se realiza mediante un flujo que ejecuta el script en un entorno de PLC.

Errores Comunes y Solución

- **Código desconocido**: Si se recibe un código que no está en `key_map`, el script simplemente lo ignora, asegurando que solo los caracteres conocidos se procesen.
- **Tecla Enter**: El código `28` es clave, ya que indica el final de una secuencia de entrada (usado aquí como el botón Enter).

Ejemplos de Uso

Ejemplo 1: Lectura de Teclado USB

1. Conecta un teclado USB al puerto USB del PLC.
2. Asegúrate de que el script `usbdrivers.py` esté en ejecución y vinculado al archivo de evento correcto (por ejemplo, `/dev/input/event0`).
3. Abre Node-RED y visualiza la salida en el nodo de depuración.

Ejemplo de Salida Esperada

Al presionar la tecla `H`, `O`, `L`, `A`, y luego Enter en el teclado USB, la salida esperada será:

```
HOLA
```

Ejemplo 2: Lector de Código de Barras

1. Conecta el lector de código de barras USB.
2. Escanea un código de barras, y el dispositivo debería enviar una serie de eventos USB.
3. El script interpretará estos eventos y mostrará el código escaneado en Node-RED.

Ejemplo de Salida

Escanear un código de barras podría generar una salida similar a esta:

```
1234567890123
```

Asegúrate de que el lector esté configurado para enviar un código de "Enter" después de cada escaneo, lo cual activará el envío de los datos a Node-RED.

Programa en Node-RED

El archivo `flows/flows_usb.json` contiene un flujo de Node-RED que ejecuta el script `usbdrivers.py` en Python para leer eventos desde dispositivos USB conectados al PLC Phoenix Contact.

Descripción del Flujo

1. **Nodo Inject**: Inicia el flujo para ejecutar el script manualmente y permite leer eventos en el dispositivo conectado.
2. **Nodo Exec**: Ejecuta el script `usbdrivers.py` en el PLC mediante el comando:

   ```
   python3 /usbdrivers/usbdrivers.py <EventNumber>
   ```

   - `<EventNumber>` representa el número de evento del puerto USB (por ejemplo, `3` o `4`).
3. **Nodo Debug**: Muestra la salida del script en la consola de depuración de Node-RED.

Ejemplo de Uso del Flujo

1. **Ejecutar el Flujo**: Abre Node-RED y ubica el flujo importado.
2. **Iniciar Lectura de Eventos**: Haz clic en el nodo `Inject` para iniciar la lectura de eventos en el dispositivo USB conectado.
3. **Ver la Salida**: La salida de los eventos USB se mostrará en el nodo `Debug` de Node-RED.

> ⚠ **Nota**: Los puertos `/dev/input/eventX` pueden variar según el dispositivo USB conectado y la configuración del sistema. Asegúrate de verificar el puerto asignado al dispositivo conectado.

Instrucciones de Configuración Adicionales

Requisitos de Sistema

- **Hardware**: Este proyecto ha sido probado en un PLC Phoenix Contact EPC 1502.
- **Sistema Operativo**: Yocto Linux Kirkstone.
- **Software Necesario**:
  - Node-RED versión 4.0.2.1 o superior, instalado en PLCnext Store.
  - Python 3.x instalado en el sistema.

Solución de Problemas Comunes

1. **Node-RED no detecta el script de Python**:
   - Verifica que la ruta a `usbdrivers.py` esté correctamente montada en el archivo `docker-compose.yml`.
2. **Permisos de Lectura de Eventos USB**:
   - Es posible que algunos eventos de `/dev/input` requieran permisos de superusuario. Añade `privileged: true` en `docker-compose.yml` para asegurarte de que Node-RED tenga los permisos necesarios.

Licencia

- **No se pueden realizar obras derivadas** basadas en este contenido.

Consulta más detalles sobre esta licencia en Creative Commons.