# Henet Switch Control Plugin for Unreal Engine

This plugin provides a simple way to monitor a serial port for switch press events from a Henet-protocol compatible hardware device and expose them to the Unreal Engine Blueprint system.

## Features

- Opens and closes a serial port connection on command.
- Listens for switch press/release events and heartbeat signals on a specified port.
- Communicates with hardware using the Henet protocol.
- Exposes events through a flexible 3-node Blueprint system.
- Runs all serial communication on a background thread to prevent any impact on game performance.

## Compatibility

This plugin is designed for **Unreal Engine 5.6**.

## Installation

1. Download the latest HenetSwitchControl-vX.X.X.zip file from the GitHub Releases page.
2. In your Unreal Engine project, create a Plugins folder at the root if it doesn't already exist.
3. Extract the downloaded zip archive into your project's Plugins folder. The structure should be YourProject/Plugins/HenetSwitchControl/.
4. Restart the Unreal Editor.
5. Open the Plugins window (Edit > Plugins) and verify that "Henet Switch Control" is enabled under the "Project" or "Installed" category.

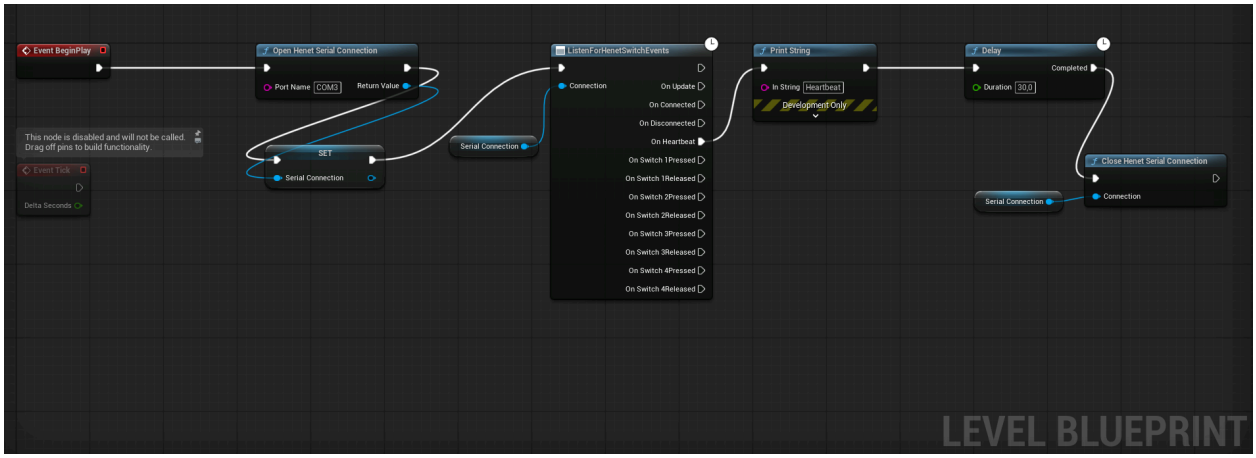
## How to Use

The plugin uses three main Blueprint nodes. Here is the standard workflow:

1. **Node 1: Open Connection**
   - In your Blueprint (like the Level Blueprint), call **Open Henet Serial Connection** on an event like Event BeginPlay.
   - Specify the **Port Name** (e.g., "COM3").
   - Drag from the blue **Return Value** pin and select **"Promote to Variable"**. Name it SerialConnection. This is very important.
2. **Node 2: Start Listener**
   - Drag off the execution pin from setting the variable and add the **Listen For Henet Switch Events** node.

- Plug your new SerialConnection variable into the Connection input.
- Plug a **self** reference into the World Context Object input.
- Connect the various output execution pins (OnConnected, OnHeartbeat, OnSwitch1Pressed, etc.) to your game logic.

3. **Node 3: Close Connection**
   - On an event like Event End Play, get your SerialConnection variable.
   - Call the **Close Henet Serial Connection** node and plug your variable into its Connection input. This is critical for preventing memory leaks and editor crashes.




# How It Works

The plugin's architecture is designed for performance and flexibility by separating the connection from the event listeners.

1. **FHenetSerialPortReader (C++ Worker)**
   - This is a C++ class running on a dedicated background thread (FRunnable).
   - It handles all low-level, blocking serial port I/O and parses the incoming Henet protocol byte stream.
   - This design ensures that reading data from the port can never stall or impact the main game thread.
2. **UHenetSerialConnection (C++ Connection Object)**
   - This is a UObject that acts as a "handle" or "reference" to the connection.
   - When created by the Open Henet Serial Connection node, it spawns the FHenetSerialPortReader worker and holds its thread-safe event queue.
   - This object is protected from garbage collection while it's active.
3. **UHenetSwitchControlLibrary (C++ Blueprint Library)**
   - This library provides the simple Open (Node 1) and Close (Node 3) functions that appear in Blueprints. They are responsible for creating and destroying the UHenetSerialConnection object.
4. **UHenetSwitchMonitorNode (C++ Async Listener Node)**
   - This is the Listen For Henet Switch Events node (Node 2).

- It is an async action (UBlueprintAsyncActionBase) that takes a UHenetSerialConnection object as input.
- It runs a timer on the game thread to poll the connection's event queue and fire the appropriate output execution pins (OnHeartbeat, OnSwitch1Pressed, etc.).
- You can have multiple, separate listener nodes all monitoring the *same* connection object.

This design keeps all blocking I/O on a background thread, while the listener(s) safely poll for events on the game thread, guaranteeing smooth performance.