Bridge Health Detector

- **Purpose**

This documentation explains how to set up a bridge health monitoring system that detects and classifies structural vibrations in real time. It utilizes the Arduino Nano 33 BLE Sense, equipped with a built-in accelerometer, to monitor movement and determine whether the bridge is in a “Normal” or “Damage” state. The results are sent instantly to a web dashboard, where authorized personnel can view the live status in real-time.

- **Workflow Overview**

The bridge health monitoring system begins by training a machine learning model using vibration data collected from the bridge structure. This model is deployed onto the Arduino Nano 33 BLE Sense, which is capable of running real-time inference on-device. During operation, the Nano continuously collects acceleration data using its onboard IMU sensor and classifies the current state of the bridge as either “Normal” or “Damage”. The classification results are transmitted over a USB serial connection to a computer. A Python Flask server running on the computer reads these predictions, timestamps them, and exposes the data through a web-accessible API. Simultaneously, a web dashboard built with HTML and JavaScript fetches this data every few seconds and displays the predictions in a color-coded table for easy visualization.

- **System Components**

The system uses the Arduino Nano 33 BLE Sense board, which includes:

- Sensor: LSM9DS1 9-axis IMU (accelerometer, gyroscope, magnetometer)
- Accelerometer Range: [-4, +4] g
- Resolution: 16-bit
- Sensitivity: -/+0.122 mg.
- Max Output Data Rate: ~952 Hz
- Communication: USB serial
- Inference: On-device ML model (Edge Impulse) classifies bridge state as “Normal” or “Damage”

This board continuously monitors structural vibrations of the bridge and makes local decisions using a trained model.

- ✓ **Arduino Firmware**

  The Arduino Nano 33 BLE Sense utilizes its built-in sensor to detect the movement of the bridge. It runs a small machine learning model to decide if the bridge is "Normal" or "Damaged." Then, it sends this result to the computer using a USB cable in a simple format.

- ✓ **Python Flask Server**

  The computer runs a Python program that listens to what the Arduino says. It reads the result, adds the current date and time, and saves it. It also shares the latest results through a web link, so a website can show them.

- ✓ **Web Dashboard**

  The website shows live results from the Arduino. It updates every 2 seconds and shows a table with the date, time, and result. If the result is "Damage," the row turns red. If it is "Normal," the row stays green.

- **Standard Operating Procedure**

To set up the bridge health monitoring system, begin by training a machine learning model using Edge Impulse and exporting it as an Arduino library. This model is then integrated into the Arduino firmware and uploaded to the Nano 33 BLE Sense. The project directory should follow a simple structure, with the main server script (server.py) placed in the root folder and the HTML dashboard (index.html) stored in a subfolder named templates. It is important to update the server.py file with the correct serial port (e.g., COM3) to match the USB port used by the Nano board.

To run the system, ensure that any serial monitor tools (such as the Arduino Serial Monitor) are closed, as they may block access to the COM port. Start the Python server by executing python server.py in the terminal. Once the server is running, open a web browser and navigate to http://127.0.0.1:5000 to access the live prediction dashboard, which displays real-time updates from the bridge monitoring system.

For regular maintenance, verify that the Nano 33 BLE Sense remains continuously powered and firmly attached to the bridge structure. If the device is disconnected or the serial port changes, restart the Python server. It is also recommended to periodically export and store the prediction logs for reporting or analysis purposes. Additionally, if a notification feature is implemented, be sure to test the alerting system routinely to ensure timely communication of any detected damage.