# Instrukcja uzytkowania: CAN Fuzzer (Random + UDS)

## 1. Wymagania

- Interfejs Vector (np. VN1610) z kanaem 0 i/lub 1
- Python 3.7+
- Biblioteka python-can (instalacja: pip install python-can)
- Obsugiwane tylko ID 11-bit lub 29-bit (w zaleznosci od ustawien)

## 2. Uruchomienie programu

- Uruchom skrypt: python combined_fuzzer_gui.py
- Pojawi sie interfejs graficzny z ustawieniami fuzzera

## 3. Pola konfiguracji

- Tryb fuzzowania: Random Fuzzing lub UDS Fuzzing
- Czas trwania [s]: ile sekund ma trwac test
- Interwa [ms]: odstep miedzy ramkami
- Minimalne ID (hex): najnizsze ID do generowania (np. 0x100)
- Maksymalne ID (hex): najwyzsze ID do generowania (np. 0x7FF)
- Extended ID: zaznacz, jesli chcesz uzywac 29-bitowych ID (tylko dla Random)
- Kana TX: kana wysyania (0 lub 1)
- Kana RX: kana odbierania (musi byc inny niz TX)

## 4. Tryby dziaania

Random Fuzzing  losowe ID, DLC, dane

UDS Fuzzing  generuje ramki z losowym SID i payloadem (do 7 bajtow)

## 5. Wyniki i logi

- Wszystkie ramki wysane  tx_log.blf
- Wszystkie ramki odebrane  rx_log.blf
- Odbierane ramki widoczne sa rowniez w oknie GUI na zywo

## 6. Przykad uzycia

- Tryb: UDS Fuzzing
- Czas: 30 sekund
- Interwa: 5 ms

- ID: 0x7E0 do 0x7EF
- TX: kana 0, RX: kana 1

Testuje diagnostyke UDS z mutowanym SID

**7. Uwagi**

- W trybie UDS nie jest uzywany extended ID
- Nie nalezy ustawiac takiego samego kanau TX i RX
- Logi mozna analizowac w CANalyzer, CANoe, Vector Log File Viewer