1. HARDWARE ARCHITECTURE OVERVIEW

The **CatSniffer** by Electronic Cats is an open-source multi-protocol wireless security assessment platform. Unlike general Software Defined Radios (SDRs), CatSniffer utilizes dedicated transceiver hardware with integrated physical MAC accelerators to capture and dissect real-time IoT frames with accurate hardware timings.

Bridge MCU: RP2040 / SAMD21

- Bridges host USB-C communication to high-speed UART.
- Controls hardware bootloader invocation and auto-reset lines.
- Enumerates as standard USB CDC serial port across all OSs.

RF Engines: TI CC1352P + Semtech SX1262

- **CC1352P:** Multi-band Sub-1GHz & 2.4GHz RF transceiver SoC.
- **SX1262:** Ultra-long range LoRa and LoRaWAN engine.
- Operates task-specific firmware images loaded via bootloader.

2. BOARD CONTROLS, PINOUT & LED STATUS INDICATORS

Buttons & Control Switches

- **RESET / RST:** Resets the RP2040 bridge MCU without power-cycling USB.
- **BOOT / FLASH:** Puts the RP2040 into UF2 mass storage bootloader mode.
- **CC_RST & CC_BSL:** Manual hardware pins to invoke the TI CC1352 serial bootloader fallback.

Status LED Indicators

- **PWR (Green/Blue):** 3.3V power rail status indicator.
- **RX / TX (Yellow/Orange):** Active UART bridge packet activity.
- **DIO / USER (Red):** Firmware-programmable indicator (blinks on successful RF packet reception).

3. CROSS-PLATFORM HOST ENVIRONMENT SETUP

Linux / macOS Configuration

Grant serial access without root permissions:

```
# Add user to dialout group (Linux) & verify:
sudo usermod -aG dialout $USER
ls -l /dev/ttyACM* /dev/ttyUSB*
```

Windows Configuration

Detect port in **Device Manager > Ports (COM & LPT)**:

```
# Enumerates as standard USB Serial Device:
COM3  (or COM4, COM5, etc.)
# Install Windows-specific binary wheel:
pip install python-magic-bin pycatsniffer
```

Python Tooling & Helper Utilities

```
# Clone official tools and install cross-platform dependencies
git clone https://github.com/ElectronicCats/CatSniffer-Tools.git
cd CatSniffer-Tools
pip install pyserial intelhex pycatsniffer
```

4. UNDERSTANDING FIRMWARE PERSONALITIES

The CatSniffer's operational mode is determined by the active firmware image loaded onto the **TI CC1352P**. To switch between sniffing **Zigbee/ 802.15.4**, **Bluetooth Low Energy (BLE)**, or **Sub-1 GHz**, you flash the corresponding firmware binary to the CC1352P.

5. THE 3 BOOTLOADER FLASHING MODES

Mode 1: Automated Serial Flashing (Standard)

Modern CatSniffer tools (`pycatsniffer` or `cc2538-bsl.py`) command the RP2040 bridge to automatically toggle the CC1352 bootloader pins over USB:

```
# Linux / macOS:
python3 cc2538-bsl.py -e -w -v -p /dev/ttyACM0
firmware.hex
# Windows:
python cc2538-bsl.py -e -w -v -p COM3 firmware.hex
```

Mode 2: RP2040 UF2 USB Flashing

Used to update the USB-to-UART bridge controller firmware:

1. Hold the **BOOT** button while connecting USB.
2. The device mounts as a mass-storage drive (`RPI-RP2`).
3. Drag and drop the latest `.uf2` bridge firmware.

Mode 3: Manual CC1352 Hardware Fallback Mode

If automated flashing fails or custom firmware locks the serial bus, manually force the CC1352 into ROM bootloader mode:

1. Hold down the **CC_BSL** button.
2. Press and release the **CC_RST** button while continuing to hold `CC_BSL` .
3. Release the **CC_BSL** button. The CC1352 ROM bootloader is now active and ready for `cc2538-bsl.py` .

6. FIRMWARE SELECTION MATRIX

Target Protocol	Required Firmware Binary	Host Capture Software
Zigbee / 802.15.4 / Thread	`sniffer_fw_CC1352P_7_v1.10.hex`	Kismet, Wireshark, TI SmartRF Packet Sniffer 2
BLE 4.x / 5.x (Sniffle)	`sniffle_cc1352p_v1.x.hex`	Sniffle (`sniff_receiver.py`), Wireshark
LoRa / LoRaWAN	`lora_raw_sniffer_sx1262.hex`	`pycatsniffer` , ChirpStack, Custom Python Script
Sub-1GHz Proprietary	TI SmartRF Studio Generic Firmware	TI SmartRF Studio, Universal Radio Hacker (URH)

7. WALKTHROUGH A: ZIGBEE & 802.15.4 MESH CAPTURE

Step 1: Flash Zigbee Firmware

```
# Flash 802.15.4 packet sniffer image to CC1352:
python cc2538-bsl.py -e -w -v -p /dev/ttyACM0 (or COM3) firmware/sniffer_fw_CC1352P_7.hex
```

Step 2: Live Ingestion via Kismet (Linux/WSL) or SmartRF 2 (Windows)

Zigbee channels 11–26 run across 2.4 GHz. **Channel 25 (2475 MHz)** avoids heavy Wi-Fi channel 1/6/11 noise:

```
# Option 1 (Linux / WSL2): Live Kismet packet capture
kismet -c "catsniffer_zigbee:device=/dev/ttyACM0,name=CatSniffer,channel=25"

# Option 2 (Windows Native GUI): TI SmartRF Packet Sniffer 2
# Select CC1352 Device -> Protocol: IEEE 802.15.4 -> Channel 25 -> Forward to Wireshark (Pipe: \.\pipe\wireshark)
```

Step 3: Convert Logs to Wireshark PCAP-NG

```
kismetdb_to_pcap --in capture.kismet --out zigbee_dissection.pcapng
```

8. WALKTHROUGH B: BLE SNIFFING WITH SNIFFLE

Step 1: Flash Sniffle Firmware Binary

```
python cc2538-bsl.py -e -w -v -p /dev/ttyACM0 (or COM3) firmware/sniffle_cc1352p.hex
```

Step 2: Stream Live Advertising & Data Frames

```
# Capture on primary advertising Channel 37 (2402 MHz) and pipe to PCAP:
sniff_receiver.py -s /dev/ttyACM0 (or COM3) -c 37 -p ble_stream.pcap

# Sniff specific MAC address and follow connection hops:
sniff_receiver.py -s /dev/ttyACM0 (or COM3) -m AA:BB:CC:DD:EE:FF -p ble_targeted.pcap
```

9. WALKTHROUGH C: WIRESHARK DECRYPTION SETUP

Zigbee Mesh Decryption

1. Navigate to `Edit > Preferences > Protocols > Zigbee`.
2. Check *"Trust Center link key required"*.
3. Add standard Zigbee HA Link Key:
   `5A:69:67:42:65:65:41:6C:6C:69:61:6E:63:65:30:39`
4. If pairing was captured, add dynamic Network Transport Key.

Bluetooth LE Decryption

1. Open `Edit > Preferences > Protocols > Bluetooth LE/SMP`.
2. Add known Long Term Keys (LTK) or static passkeys.
3. Useful display filters:
   - `btle.advertising_address`
   - `btatt.opcode` (GATT read/write)
   - `btle.data_header.length > 0`

10. PRACTICAL IOT & RF ASSESSMENT USE CASES

Smart Home Mesh Auditing (Zigbee / Matter)

Auditing home automation ecosystems (smart bulbs, motion sensors, door locks, bridges):

- Capture unencrypted association requests during initial device provisioning.
- Audit default Trust Center Link Keys across commercial Zigbee smart hubs.
- Map network topology, router node routes, and endpoint clusters in Kismet.

BLE Peripheral & GATT Reverse Engineering

Auditing fitness trackers, medical telemetry, smart tags, and smart locks:

- Capture raw advertisement packets on channels 37, 38, and 39.
- Map exposed GATT services and inspect unencrypted sensor data in transit.
- Audit pairing mechanisms (Just Works vs Passkey Entry) during connection handshakes.

LPWAN & Long-Range Telemetry (LoRa)

Auditing smart utility meters, industrial sensors, and remote SCADA gateways:

- Sniff raw Sub-1 GHz ISM band transmissions (433/868/915 MHz) via SX1262.
- Inspect LoRaWAN Join Requests, DevEUI, AppEUI, and unencrypted payloads.
- Audit packet integrity and frame counters across LPWAN telemetry nodes.

Industrial & Proprietary Sub-1 GHz RF

Auditing AMR meters, garage remotes, gate controllers, and Amazon Sidewalk:

- Demodulate FSK/OOK/GFSK signals using TI SmartRF Studio & URH.
- Analyze rolling code behaviors and replay vulnerability surfaces.
- Assess enterprise asset tracking beacons and long-range utility transmitters.

11. OPERATIONAL RULES OF ENGAGEMENT & SAFETY

- **SMA Antenna Safety:** Always screw in the 50Ω SMA antenna before powering the unit. Transmitting or receiving without an antenna can degrade sensitive RF front-end circuitry.
- **Wi-Fi Spectrum Coexistence:** Standard Wi-Fi channels 1, 6, and 11 saturate 2.4 GHz. For the cleanest Zigbee/Thread packet captures, lock sniffing to **Channel 25 (2475 MHz)** or **Channel 26 (2480 MHz)**.
- **Ethical Auditing:** Only capture and dissect RF telemetry on networks, devices, and frequency bands you own or have explicit written authorization to assess.

12. OFFICIAL RESOURCES & COMMUNITY LINKS

Resource	Location / Command
Electronic Cats Official Repo	`https://github.com/ElectronicCats/CatSniffer`
Python CatSniffer Library	`pip install pycatsniffer`
Sniffle BLE Sniffer Suite	`https://github.com/nccgroup/Sniffle`
TI SmartRF Packet Sniffer 2	`https://www.ti.com/tool/PACKET-SNIFFER` (Native Windows)
CC2538 Serial Flasher Utility	`https://github.com/JelmerT/cc2538-bsl`

13. LIVE REAL-TIME WIRESHARK STREAMING (NAMED PIPES)

Instead of saving capture files to disk and opening them afterward, stream live IoT frames directly into Wireshark in real time:

Linux / macOS (FIFO Pipe)

```
# 1. Create a named pipe:
mkfifo /tmp/catpipe
# 2. Launch Wireshark listening on pipe:
wireshark -k -i /tmp/catpipe &
# 3. Stream Sniffle BLE packets:
sniff_receiver.py -s /dev/ttyACM0 -c 37 -o /tmp/catpipe
```

Windows (Named Pipe Interface)

1. In **TI SmartRF Sniffer 2**, enable *Pipe output*: `\.\pipe\wireshark`.
2. In Wireshark, go to **Capture > Options > Manage Interfaces > Pipes**.
3. Add `\.\pipe\wireshark` and click **Start**.

14. ANTENNA SELECTION & FREQUENCY MATCHING

The CatSniffer features a single 50Ω SMA connector, but RF antennas are strictly tuned to specific physical wavelengths:

- **2.4 GHz Antenna (Short Duck):** Required for **Zigbee, Thread, Matter, and BLE**. Using a Sub-1GHz antenna will severely attenuate 2.4 GHz signals.
- **Sub-1 GHz Antenna (Tuned to 915 MHz, 868 MHz, or 433 MHz):** Required for **LoRa, LoRaWAN, and proprietary ISM remotes**. Standard Wi-Fi antennas cannot resonate on Sub-1GHz bands.

15. TROUBLESHOOTING: THE LINUX "MODEMMANAGER" SERIAL CONFLICT

On Debian/Ubuntu Linux systems, the background daemon `ModemManager` automatically grabs newly connected USB-UART devices (RP2040/SAMD21) to probe them as cellular modems, causing `Device or resource busy` errors when flashing:

```
# Temporary fix: Stop the service before flashing
sudo systemctl stop ModemManager
# Permanent fix: Create udev rule (/etc/udev/rules.d/99-catsniffer.rules)
ATTRS{idVendor}=="2e8a", ENV{ID_MM_DEVICE_IGNORE}="1"
```

16. ZERO-COST PRACTICE TARGETS (START SNIFFING IMMEDIATELY)

Smartphone BLE Beacon Target

Install the free **nRF Connect for Mobile** app (iOS/Android). Go to the **Advertiser** tab to broadcast custom BLE beacons, heart rate telemetry, or battery levels, then sniff and decode them live with CatSniffer.

Affordable Smart Home Targets

Standard smart plugs (such as IKEA TRÅDFRI or Sonoff Zigbee sensors) offer immediate, low-cost IEEE 802.15.4 join handshakes and periodic beacon frames for isolated lab practice.

17. WIRESHARK PROTOCOL DISPLAY FILTER QUICK-REFERENCE

Protocol Target	Wireshark Display Filter	Frame Details Isolated
Zigbee Network	`zbee_nwk`	Mesh routing, device joins, route discovery, and PAN beacons
Zigbee Application	`zbee_aps`	Application profiles, cluster commands (on/off, brightness, sensor state)
Raw 802.15.4 MAC	`wpan`	Frame control headers, sequence numbers, addressing, PAN IDs
Thread / 6LoWPAN	`6lowpan` or `thread`	IPv6 datagram compression, mesh headers, and UDP telemetry
BLE Advertisements	`btle.advertising_address`	Broadcast beacons, manufacturer specific payload data
BLE GATT Attributes	`btatt.opcode`	GATT read/write requests, responses, and real-time notifications

PORT::ZERO CYBER SOLUTIONS
Protect Against Cyber Threats ®